RELIABILITY MODELING OF DIGITAL CONTROL SYSTEMS USING THE MARKOV/CELL-TO-CELL MAPPING TECHNIQUE

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of The Ohio State University

By

Diego Mandelli, Laurea

*****

The Ohio State University
2008

Master's Examination Committee:

Dr. Tunc Aldemir, Advisor

Dr. Don Miller

Approved by

____

Advisor
Graduate Program in Nuclear Engineering

ABSTRACT

Many nuclear power plants are replacing aging analog instrumentation and control (I&C) systems with modern digital systems. In several plants, non-safety related systems, such as feedwater control, are already controlled by digital equipment. The replacement of an existing component with a new component may affect the safety and the reliability of the overall system. In particular, this is valid if the component added, such as a digital control system, has different failures modes compared to an analog control system and these failure modes affect the behavior of the overall system differently.

In the reliability modeling of digital control systems, conventional approaches based on the event-tree/fault-tree (ET/FT) methodology have limited capabilities in the representation of the statistical dependence between failure events. Dynamic methodologies can be considered as an important alternative to overcome this limitation and the Markov/CCMT (cell-to-cell mapping technique) has been proposed as an alternative dynamic methodology for the probabilistic risk assessment (PRA) of digital control systems.

In this thesis, the Markov/CCMT is illustrated using the digital control system of the feedwater system of a pressurized water reactor (PWR). Discrete hardware/software/ firmware states are defined and transitions between these states are deduced from the control logic of the system, as well as from the failure modes and effects analysis performed on each component. The CCMT is used to represent the dynamics of the system in terms of probability of transitions between process variable magnitude intervals (cells) that partition the state space. The resulting event sequences is converted into dynamic event trees (DETs) which can be incorporated into an existing ET/FT based PRA of a PWR using an existing PRA tool such as the SAPHIRE code.

dedicato a tutte le persone che hanno vissuto con me questo sogno

dedicated to all the people who have lived this dream with me

ACKNOWLEDGMENTS

I wish to thank Dr. Aldemir for his intellectual support during these years and his guidance and enthusiasm which made this thesis possible.

I am also grateful to Dr. Miller, Dr. Ekici, Dr. Bucci, Jason Kirschenbaum and Dr. Stovsky for their assistance and stimulating discussions.

I want to say thanks to my family which helped me a lot in these years away from them.

Finally I want to thank Cheri for the moral support and the encouragement

VITA

1978	Born in Brescia, Italy
1997	High School Degree, Electronics and Telecommunications Technical degree, Brescia, Italy
2004	Polytechnic of Milan, Italy: Laurea in Nuclear Engineering Final degree score of 92/100 Thesis: “Study on pressurized water reactor cores for space applications”
2007	The Ohio State University: Master in Nuclear Engineering

PUBLICATIONS

Research publications

1. “An Event Tree/Fault Tree/Embedded Markov Model Approach for the PSAM-8 Benchmark Problem Concerning a Phased Mission Space Propulsion System”, D. Mandelli, T. Aldemir, PSAM8: Proceedings of the 6th International Conference on Probabilistic Safety Assessment and Management, New Orleans 2006.

2. ”A Benchmark System for the Assessment of Reliability Modeling Methodologies for Digital Instrumentation and Control Systems in Nuclear Plants“, T. Aldemir, D. Mandelli et al., Proceedings NPIC&HMIT 2006, Albuquerque, New Mexico (2006).

3. “Incorporation of Markov Reliability Models for Digital Instrumentation and Control Systems into Existing PRAs”, T. Aldemir, D. Mandelli et al., Proceedings NPIC&HMIT 2006, Albuquerque, New Mexico (2006).

4. “Reliability Modeling of Digital Instrumentation and Control Systems for nuclear Reactor Probabilistic Risk Assessments”, NUREG/CR-6942, U. S. Nuclear Regulatory Commission, Washington, D.C. (2006).

5. “A Risk-Informed Approach in the Design of a Molten Salt Reactor,” T. Aldemir, D. Mandelli et al., ANS 2007, Boston, Massachusetts (2007).

6. “A Benchmark System for the Reliability Modeling of Digital Instrumentation and Control Systems, D. Mandelli, T. Aldemir et al., PSAM9: Proceedings of the 7th International Conference on Probabilistic Safety Assessment and Management, Hong Kong 2008.

7. “Markov/CCMT Modeling of the Benchmark System and Incorporation of the Results Into an Existing PRA, D. Mandelli, T. Aldemir et al., PSAM9: Proceedings of the 7th International Conference on Probabilistic Safety Assessment and Management, Hong Kong 2008.

FIELDS OF STUDY

Major Field: Nuclear Engineering

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter discusses the possible impact of digital control systems on nuclear power plant operation from a reliability view point and, thus, the motivation for this thesis. The logic structure of this work is then shown in Section 1.3

1.1 Toward Digital Communication and Control Systems (I&C)

Instrumentation and control (I&C) systems are widely used in nuclear power plants for monitoring, control and protection [1]. Since the beginning of the nuclear era in the 1940s, analog systems have accomplished these tasks satisfactorily. Although there are some design issues, such as inaccurate design specifications and susceptibility to certain environmental conditions, the primary concerns with the extended use of analog systems are the effects of aging such as mechanical failures and environmental degradation.

Digital systems are essentially free of drift that afflicts analog systems and, thus, they maintain their calibration better. Digital systems can improve system performance due to their accuracy and computational capabilities such as:

- Self testing
- Signal validation
- Process system diagnostics
- Fault tolerance
- Higher data handling
- Storage capabilities.

In this respect, nuclear power plants are in the process of replacing and upgrading aging and obsolete I&C systems with digital ones. The replacement of an existing system with a new component may affect the safety and the reliability of the overall system. This is particularly valid if the component added, such as a digital control system, has different failures modes compared to an analog control system and these failure modes affect the behavior of the overall system differently.

Probability risk assessment (PRA) is a commonly used tool not only in the nuclear but also the chemical and the aeronautical industries to examine the safety and reliability of specific systems. Due to the nature of digital systems, conventional PRA tools, such as fault trees and event trees (FT and ET), may not possess the capability to model most digital systems since these tools have limited capabilities in the representation of the statistical dependence between failure events. Dynamic methodologies are those that explicitly account for the time element in system evolution and have been proposed as alternatives to conventional tools for the reliability modeling of digital systems. The NUREG/CR-6901(Current State of Reliability Modeling Methodologies for Digital

Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments) [2] contains a recent survey of these methodologies

In 2002 the U.S. Nuclear Regulatory Commission (NRC) initiated a collaborative study with The Ohio State University (OSU) to develop both policies and methods for inclusion of reliability models for digital systems into current generation nuclear power plant PRAs. This study identified the Markov/CCMT (Chapter 7) and dynamic flowgraph methodology (DFM) [3] as methodologies that rank as the top two with most positive features and least negative or uncertain features (using subjective criteria based on reported experience) [2].

1.2 Motivation of this work

The purpose of this thesis can be summarized in the following two points:

- show how it is possible to model digital control systems for PRA purposes using the Markov/CCMT methodology, and
- produce event sequences or dynamic event trees which can be incorporated into an existing ET/FT based PRA of a PWR using the SAPHIRE code [4].

The features of digital I&C systems that need to be accounted for in their reliability modeling and which have been captured by the Markov/CCMT methodology are the following:

- dependence of the control action on system history,
- dependence of system failure modes on exact timing of failures,
- functional as well as intermittent failures,
- error detection capability,
- possible system recovery from failure modes.

These features and others that need to be accounted in the reliability modeling if digital I&C systems for are described in [5].

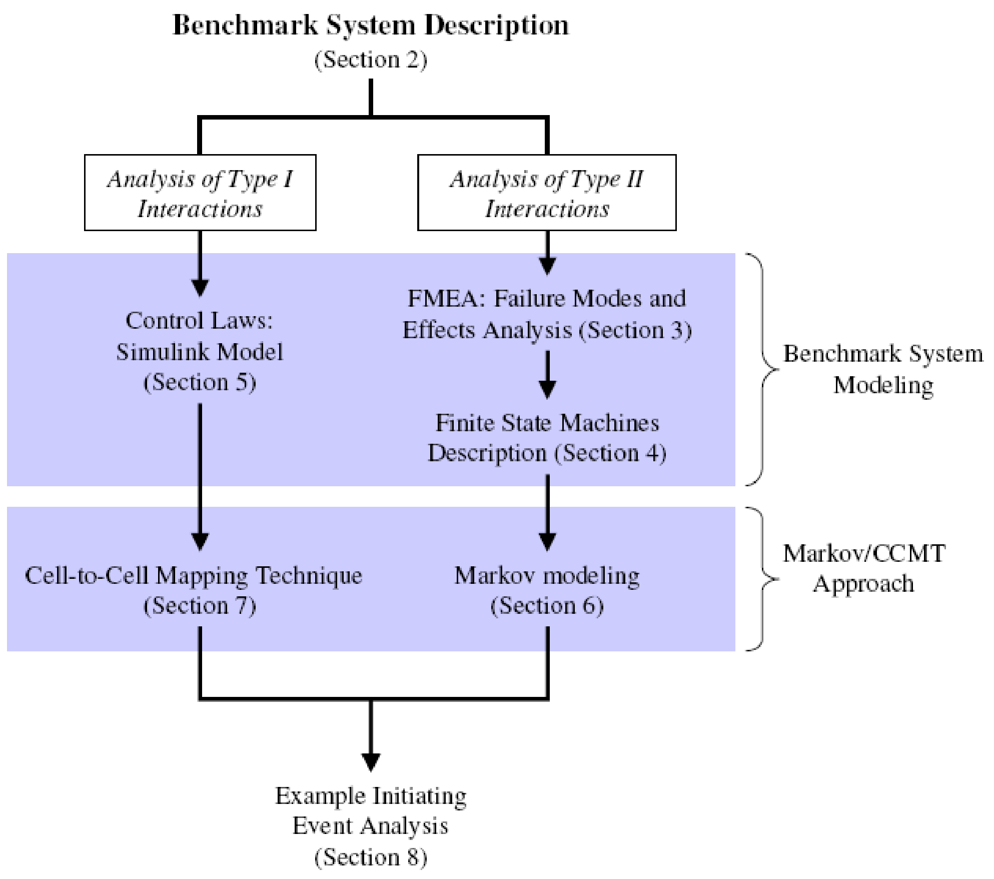
1.3 Thesis Organization

As shown in Fig.1.1 the modeling of digital control systems through the Markov/CCMT methodology consists of several steps which account for two types of interactions:

- the interactions between the digital I&C system and controlled/monitored process (defined in [2] as Type I interactions), and
- the interactions between different components of the controller itself (defined in [2] as Type II interactions),.

For each of these two analyses, the following two steps have to be performed:

- Construct a Markov model
- Apply the set of rules of the Markov/CCMT methodology



Benchmark System Description
(Section 2)
Analysis of Type I Interactions
Analysis of Type II Interactions
Control Laws: Simulink Model (Section 5)
FMEA: Failure Modes and Effects Analysis (Section 3)
Finite State Machines Description (Section 4)
Benchmark System Modeling
Cell-to-Cell Mapping Technique (Section 7)
Markov modeling (Section 6)
Markov/CCMT Approach
Example Initiating Event Analysis (Section 8)

Fig.1.1 – General layout of the work presented in this thesis

Chapter 2 describes in detail the benchmark system that is analyzed: the digital control system of a feedwater system (DFWCS) of a PWR. The components, the connections among them and the control laws are presented in detail.

The modeling of the Type II interactions is introduced in Chapters 3 and 4. Chapter 3 shows the failure modes and effects analysis (FMEA) for all the components of the DFWCS and Chapter 4 implements the information contained in Chapter 3 using a finite state machine description for each component.

The modeling of Type I interactions is described in Chapter 5 thorough a Simulink model which implements the control laws and the effects of possible failures modes of the DFWCS components on the dynamic of the system.

Chapters 6 and 7 show how the information obtained from system modeling is used to perform the Markov/CCMT analysis. In particular, Chapter 6 shows how it is possible to model Type II interactions thorough a Markov transition diagram. Markov transitions diagrams are deduced from the finite state machine descriptions presented in Chapter 4 in order to describe the temporal behavior of the controller’s components. Chapter 7 shows how Type I interactions are modeled thorough the CCMT. This technique is used to represent the dynamics of the system in terms of probability of transitions between process variable magnitude intervals (cells) that partition the state space.

Finally, Chapter 8 shows how it possible to implement the Markov/CCMT methodology on a simplified version of the benchmark system in order to obtain event sequences and dynamic event trees which can be used as input for the SAPHIRE code.

CHAPTER 2

SYSTEM DESCRIPTION

This chapter introduces the benchmark system that is used to show how it is possible to model digital I&Cs systems using Markov/CCMT. The system is described in detail and the components, connections among them and control laws are presented.

2.1 System overview

The system under consideration is the feedwater system of typical two loops PWR (see Fig.2.1) [6, 7, 8, 9]. Each steam generator (SG) has its own digital feedwater controller. The purpose of the DFWCS is to maintain the water level inside each of the SGs optimally within ± 2 inches (with respect to some reference point) of the setpoint level (defined at 0 inches). The controller is regarded as failed if water level in a SG rises above +30 inches and falls below -24 inches. Each feedwater controller is connected to three actuated devices:

- a feedwater pump (FP),
- a main feedwater regulating valve (MFV), and

- a bypass feedwater regulating valve (BFV).

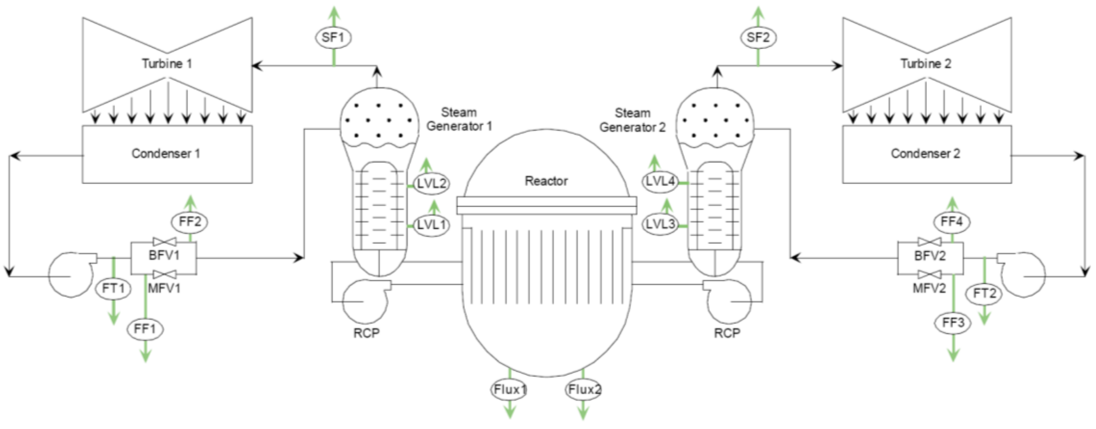


Fig.2.1 – Overall overview of the feedwater system

In the following sections, a detailed description of the DFWCS (components, connection and control logic) is presented.

2.1.1 Actuated devices

The controller regulates the flow of feedwater to the steam generators to maintain a constant water level in the own steam generator. In addition to the FP, FP seal water system, MFV, and BFV, the feedwater control system contains high pressure (HP) feedwater heaters and associate piping and instrumentation [6, 7, 8, 9].

Feed pumps are steam turbine driven, horizontal, double-suction, double volute, single stage, centrifugal pumps. The pumps have a design output of 15,000 gpm at a suction rate of 318.7 psia and a discharge pressure of 118.9 psia. The normal operating discharge pressure is approximately 1100 psig at 100%. The FP is driven by a dual admission, horizontal, 9140 HP, 5350 rpm steam turbine. During plant operation with power greater than 5%, the turbine is aligned to the reheat and main steam system. Steam is supplied from the main steam system during plant startup until reheat steam pressure is sufficient to supply the turbines. If main steam is not available or power is less than 5%, steam can be supplied to the feed pump turbine from the auxiliary steam system. The purpose of the FPs is to pump the feedwater through the high pressure feedwater heaters into the SGs with sufficient pressure to overcome both the SG secondary side pressure and the frictional losses between the feed pump and the SG inlet.

The MFV and BFV regulate the amount of feedwater going into the SG in order to maintain a constant water level in the SG. The MFV is a 10 inch, air operated, angle control valve with 16 inch end connections. This valve is made of steel and has a design rating of 2160 psig at 1000°F. The actuator is a piston type actuator, with separate instrument air supplies to the top and the bottom of the piston. Ball valves control the admission of operating air to the piston for opening and closing operations. The BFV is a 6 inch, air operated, steel control valve.

2.1.2 Operating modes and control logic

The DFWCS operates in different modes depending on the power generated in the primary system. These modes are the following [6, 7, 8, 9]:

- Low power mode
- High power mode
- Automatic transfer from low to high power mode
- Automatic transfer from high to low power mode

Low power mode of operation occurs when the reactor operates between 2% and 15% reactor power. In this mode, the BFV is used exclusively to control the feedwater flow. The MFV is closed and the FP is set to a minimal speed. The control laws use the feedwater flow, feedwater temperature, feedwater level in the steam generator, and neutron flux to compute the BFV position. The feedwater level is fed to a proportional integral differential (PID) controller using the feedwater temperature to determine the gain. Then this value is summed with the feedwater flow and neutron flux. Essentially, neutron flux and feedwater flow are used to predict required changes in water levels.

High power mode is used when the reactor power is between 15% and 100% reactor power. In this mode, the MFV and the FP are used to control the feedwater flow. The BFV is closed in a manner that is similar to low power mode. The control laws (see Section 2.2.2) use the feedwater level in the steam generator, steam flow, and feedwater flow to compute the total feedwater demand. This computed value is used to determine both the position of the MFV and the speed of the FP. The FP also uses the other digital feedwater MFV controller's output to compute the speed needed. The feedwater flow and steam flow are summed and fed to a set of PI controller algorithms. The output from these controller algorithms is added to the feedwater level and that result is fed to a PI controller algorithm that uses the steam flow for the controller algorithm's gain.

Each digital feedwater controller is comprised of several components (see Fig.2.2 and Fig.2.3) which provide both control and fault tolerant capabilities. The control algorithms are executed on both a main computer (MC) and backup computer (BC). These computers produce both analog and digital output signals for the MFV, BFV, FP and pressure differential indicator (PDI) controllers, as shown in Fig.s 2.2 and 2.3, respectively.

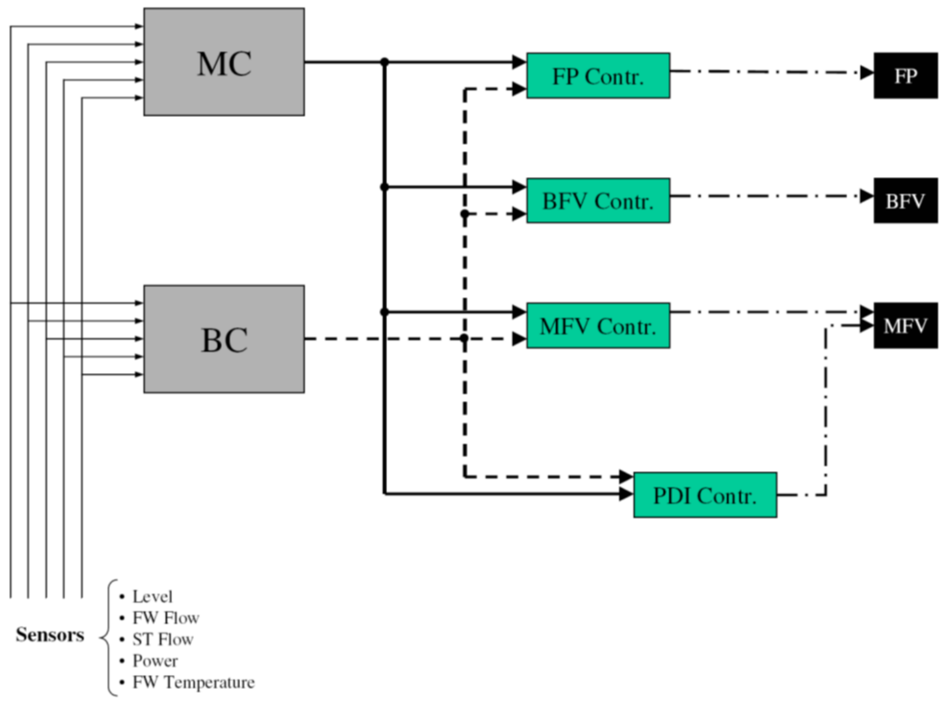


Fig.2.2 – Analog connections for the DFWCS control system

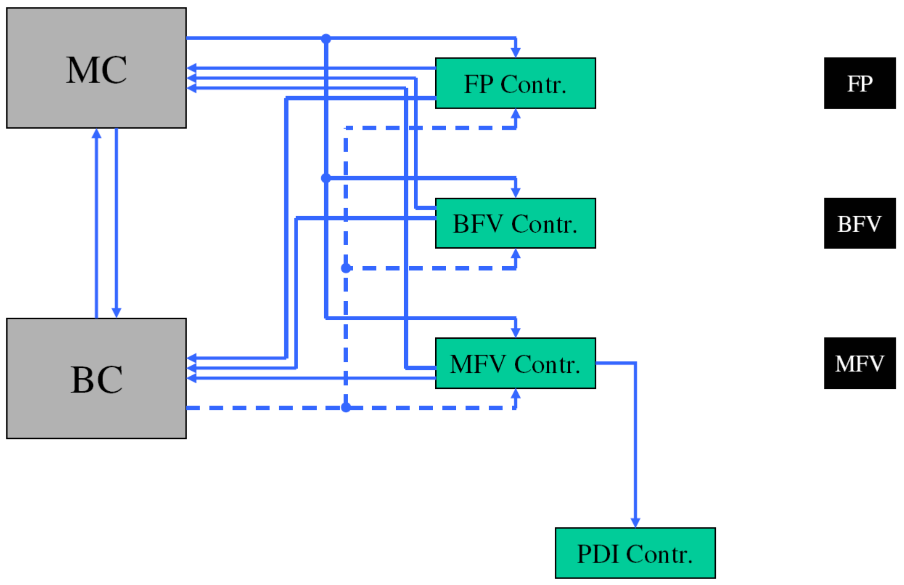


Fig.2.3 – Digital connections for the DFWCS control system

Each of these controllers forward the MC or BC's outputs to their respective controlled device (MFV, BFV or FP), or it can maintain the previous output to that device. If the controllers decide to maintain a previous output value to a controlled device, it is necessary for operators to override the controller.

Transitions between low and high power are controlled by the neutron flux readings. When the system is in low power mode and the neutron flux increases to a point at which high power mode is necessary, the MFV is signaled to open while the BFV closes to maintain needed feedwater flow. The opposite situation occurs when the system is in high power mode and the neutron flux decreases to a point when low power mode is needed.

2.2 Detailed view of the benchmark system

This section describes the DFWCS at a greater level of detail [6, 7, 8, 9]. In particular, the physical connections between the sensors, computers, controllers and actuated devices (MFV, BFP and FP) are examined. In addition, the control laws are stated and the fault tolerant features of the architecture are described.

2.2.1 Physical connections of the DFWCS

The DFWCS obtains information about the state of the controlled process through the use of several sensors that measure (see Fig.2.1):

- feedwater level,
- neutron flux,
- feedwater flow,
- steam flow, and
- feedwater temperature.

As shown in Fig.2.4, the sensor signals are routed to provide information to both the MC and BC. The set point data is delivered from the MFV controller to the MC and BC through an analog signal.

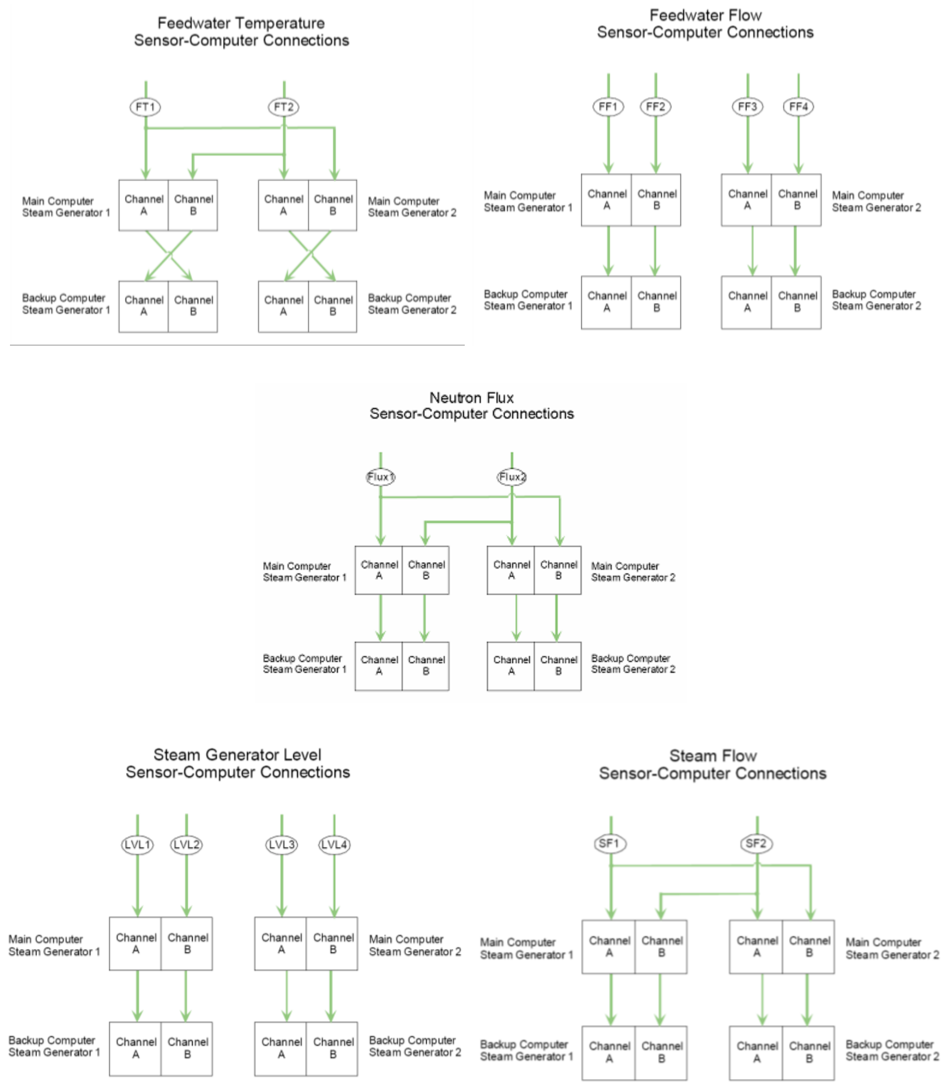
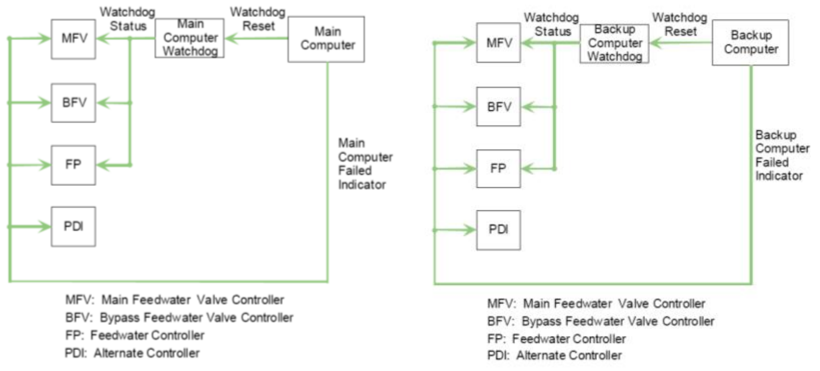

Feedwater Temperature Sensor-Computer Connections
FT1
FT2
Main Computer Steam Generator 1
Main Computer Steam Generator 2
Backup Computer Steam Generator 1
Backup Computer Steam Generator 2
Channel A
Channel B
Feedwater Flow Sensor-Computer Connections
FF1
FF2
FF3
FF4
Neutron Flux Sensor-Computer Connections
Flux1
Flux2
Steam Generator Level Sensor-Computer Connections
LVL1
LVL2
LVL3
LVL4
Steam Flow Sensor-Computer Connections
SF1
SF2

Fig.2.4 – Connection schemes for the computers (MC and BC) and the sensors

The DFWCS components are connected together in several different ways as shown in Fig.2.2 and Fig.2.3. First, both the MC and BC provide input signals to the MFV, BFV and FP controllers through an analog control signal and failure status signals. The MFV, BFV, and FP controllers are configured within the DFWCS to share status information. The PDI controller serves as a backup for the MFV controller by sampling the output of the MFV controller. If the MFV controller output is lost, the PDI will send the last good MFV controller signal to the MFV. The PDI controller also shares status information with the MFV, BFV and FP controllers.

Figure 2.5 shows the connections between computer, controller and the watchdog timer. A watchdog timer is a hardware timer used to determine if a software error or other computer failure has rendered a processor unusable. A normally functioning computer resets the watchdog timer at regular intervals. However, in the presence of a software error or another computer failure, the timer will not be reset by the computer and the timer can go off. For example, a runaway process, halted (failed) processor, or a sufficiently lengthy computational delay may result in failure to reset the watchdog timer. As a result, the watchdog timer may go off. If the timer goes off, all components in the controller connected to the watchdog timer are notified of the computer failure. In the case of the benchmark system, the MV, BFV, and FP controllers are notified and transfer control away from the affected computer.



Watchdog Status
Main Computer Watchdog
Watchdog Reset
Main Computer
MFV
BFV
FP
PDI
Main Computer Failed Indicator
MFV: Main Feedwater Valve Controller
BFV: Bypass Feedwater Valve Controller
FP: Feedwater Controller
PDI: Alternate Controller
Watchdog Status
Backup Computer Watchdog
Watchdog Reset
Backup Computer
MFV
BFV
FP
PDI
Backup Computer Failed Indicator
MFV: Main Feedwater Valve Controller
BFV: Bypass Feedwater Valve Controller
FP: Feedwater Controller
PDI: Alternate Controller

Fig.2.5 – Connection schemes for the computers (MC and BC), the watchdog timer and the controllers

2.2.2 Control laws

The control laws for the feedwater controller for SGn (n = 1, 2; see Fig.2.1) under normal system operation have been taken from the control algorithm of the DFWCS (written in C code) of an operating PWR and can be expressed as in Eqs. (2.1) – (2.13) below [6, 7, 8, 9]. The symbols used in Eqs. (2.1) – (2.13) are listed in Table 2.1. As mentioned in Section 2.1, the inputs of the computers are the readings of the sensors (i.e. SG level, feedwater flow, steam flow, power and feedwater temperature). The first step of the calculation determines the flow demand needed. This value is also function of the operating modes of the controller (i.e. high or low power mode).

Variable Symbol	*Variable Name*
x_n	Level [ft]
f_{wn}	Feedwater flow
f_{sn}	Steam flow
A	SG sectional area [ft^2]
C_{Ln}	Compensated water level
E_{Fn}	Compensated flow error
p_n	Compensated power
C_{pn}	Power
C_{Fn}	Flow demand (High Power)
C_{Bn}	Flow demand (Low Power)
r_n	Level set point
σ_{Mn}	MFV Demand
σ_{Bn}	BFV Demand
σ_{Fn}	FP Demand
$\tilde{S}_{Fn}(t)$	FP position
$\tilde{S}_{Mn}(t)$	MFV position
$\tilde{S}_{Bn}(t)$	BFV position

Table 2.1 – List of symbols used in the controller algorithm

Rate of level change: $$\frac{dx_n}{dt} = A \cdot (f_{wn} - f_{sn}) \quad (2.1)$$

Compensated water level: $$\tau_2 \frac{dC_{Ln}}{dt} = -C_{Ln} + x_n + \tau_1 \cdot (f_{wn} - f_{sn}) \quad (2.2)$$

Compensated flow error: $$\tau_6 \frac{dE_{Fn}}{dt} + E_{Ln}(t) = \tau_7 \cdot (\frac{df_{wn}}{dt} - \frac{df_{xn}}{dt}) \quad (2.3)$$

Compensated power: $$\tau_4 \frac{dC_{pn}}{dt} = -C_{pn}(t) + p_n + \tau_3 \frac{dp_n}{dt} \quad (2.4)$$

Flow demand (High Power):

$$C_{Fn}(t) = \beta_{Fn}(f_{sn}) \int dt[r_n - C_{Ln}(t) + E_{Fn}(t)] - \lambda_{Fn}(\sigma_{Bn}) \quad (2.5)$$

Flow Demand (Low Power):

$$C_{Bn}(t) = \nu_{Bn}\alpha_M + \nu_{Bn}C_{Pn} + \beta_{Bn}(h_{wn}) \int dt[r_n - C_{Ln}(t)] - \lambda_{Mn}(\sigma_{Mn}) \quad (2.6)$$

Value of the flow demand (*C_{Fn}* or *C_{Bn}*) is then translated into demand, position (*σ_{Bn} σ_{Mn}* for BFV and MFV respectively) or speed (*σ_{Fn}* for FP), for all the three actuated devices (MFV, BFV, FP). This is done through lookup tables as shown in Fig.2.6 for the MFV and FP. Flow demand is a normalized valued which can range in the interval 0-100, valve position and pump speed are values located in the interval 1-10.

FP Demand: $$\sigma_{Fn}(t) = \begin{cases} \sigma_{Fn}(\mathrm{C_{Fn}} = 0) & \text{(Low Power)} \\ \sigma_{Fn}(\max(\mathrm{C_{Fn}}, \sigma_{Mn}^{-1}(\mathrm{C_{Fn}}))) & \text{(High Power)} \end{cases} \quad (2.7)$$

MFV Demand: $$\sigma_{Mn}(t) = \begin{cases} 0 & \text{(Low Power)} \\ \sigma_{Mn}(\mathrm{C}_{\mathrm{Fn}}) & \text{(High Power)} \end{cases} \tag{2.8}$$

BFV Demand: $$\sigma_{Bn}(t) = \begin{cases} C_{Bn}(t) & \text{(Low Power)} \\ 0 & \text{(High Power)} \end{cases} \tag{2.9}$$

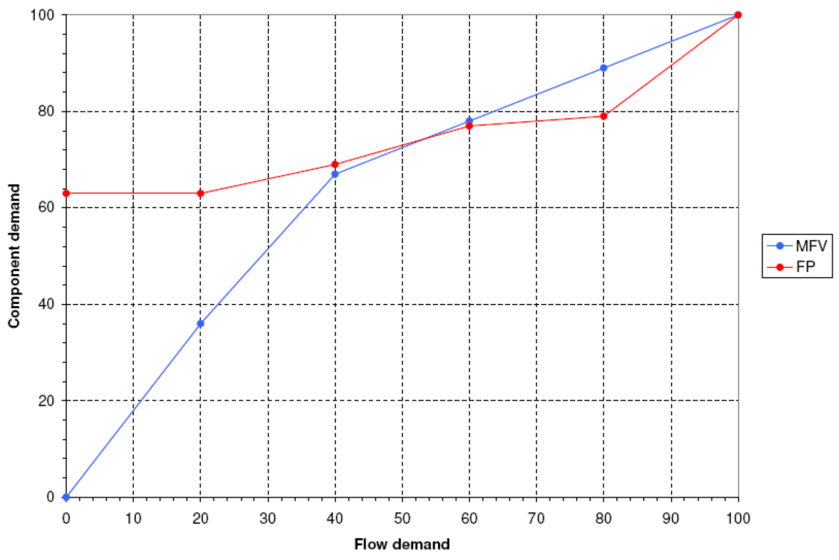


Fig.2.6 – Lookup table for the MFV and FP

Finally, the value of the actuated device demand is translated into speed or position. At the position, the status of the MC and BC computers (i.e., computer operating or computer down) is taken into account.

FP Speed: $$\tilde{S}_{Fn}(t) = \begin{cases} \sigma_{Fnm} & \text{MC operating} \\ \sigma_{Fnb} & \text{BC operating, MC down} \\ \eta_{Fn} & \text{BC down, MC down} \end{cases} \tag{2.10}$$

$$\text{MFV Position:} \qquad \tilde{S}_{Mn}(t) = \begin{cases} \sigma_{Mnm} & \text{MC operating} \\ \sigma_{Mnb} & \text{BC operating, MC down} \\ \eta_{Mn} & \text{BC down, MC down} \end{cases} \tag{2.11}$$

$$\text{BFV Position:} \qquad \tilde{S}_{Bn}(t) = \begin{cases} \sigma_{Bnm} & \text{MC operating} \\ \sigma_{Bnb} & \text{BC operating, MC down} \\ \eta_{Bn} & \text{BC down, MC down} \end{cases} \tag{2.12}$$

As mentioned in Section 2.1.2, the PDI controller enters into action in case of a loss of communication between MFV controller and MFV. The PDI decision logic can be expressed as the following:

$$\text{PDI Decision:} \qquad S_{Fn}(t) = \begin{cases} 0 & \tilde{S}_{Mn} > 0 \\ \eta_{Bn} & \text{Otherwise} \end{cases} \tag{2.13}$$

Also, all sensor inputs are averaged before being processed by the control laws. For example, the feedwater level for SG1 is the average of the two feedwater level sensors LV1 and LV2 (see Fig.2.1). Equations 2.2 - 2.4 compute the flow demand for high power mode for the feedwater controller. Equation 2.6 computes the BFV demand for low power mode. The dynamic gain $\beta_{Bn}(h_{wn})$ and $\lambda_{Mn}(\sigma_{Mn})$ in Eq. 2.6 obtained from a lookup table on the feedwater temperature and the MFV opening respectively. The subscripts *m* and *b* in Eqs. 2.10-2.13 refer to signals from the main and backup CPUs respectively. The η_{Fn} , η_{Mn} and η_{Bn} in Eqs. 2.10-2.13 denote history data for the FP, MFV and BFV positions, respectively. If both the MC and the BC have failed, these data are used to determine the FP, MFV and BFV positions.

2.2.3 Fault tolerant features

Since the MFV, BFV and FP controllers forward the control signals to the corresponding control points (the MFV, BFV, and FP, respectively, as well as the PDI controller), they provide a level of fault tolerance [6, 7, 8, 9] if both the MC and BC fail by allowing the operators time to intervene by holding the outputs of each to a previously valid value.

The MC and BC, the MFV, BFV and FP and the PDI controllers are each connected to an independent power source wired to a separate bus. A single power source failure can only affect one computer, all of the MFV/BFV/FP controllers, or the PDI controller at one time.

Both the MC and BC are set to oversample at 3 times the Nyquist criterion (the Nyquist criterion states that the highest frequency present in a signal must be less than half of the sample frequency) to avoid aliasing. Moreover, a failure in the MC or BC can be detected and the fail over (fail over is the process in which a degraded component is removed from control and replaced by a healthy component) to a healthy component can occur with enough time to meet the response requirements of the process.

The water level set point is taken from a switch connected to the MFV and is propagated to both the MC and BC. If the set point signal goes out of range, then the computers fall back on a preprogrammed set point value.

Each computer (MC or BC) is connected to a watchdog timer. A watchdog timer is a hardware timer and associated connections used to determine if a software error or other computer failure has rendered a processor unusable. A normally functioning computer resets the watchdog timer at regular, defined intervals so the timer does not “go off.” However, in the presence of a software error or another computer failure, the timer will not be reset by the computer and the timer can go off. For example, a runaway process, halted (failed) processor, or a sufficiently lengthy computational delay may result in failure to reset the watchdog timer. As a result, the watchdog timer may go off. If the timer goes off, all components in the controller connected to the watchdog timer are notified of the computer failure. In the case of the benchmark system, the MFV, BFV, and FP controllers are notified and transfer control away from the affected computer.

Each computer (MC or BC) verifies and validates its inputs, checking for out range and excessive rate changes in the inputs that would indicate errors in the sensor readings or problems with the analog to digital conversion of the values. Each computer will ignore input that fails these checks if the other inputs are still valid.

Deviation between the two sensors is detected and, if the deviation is large enough, the computer can signal a deviation error to the MFV, BFV, and FP controllers so they may switch to the other computer.

The PDI controller provides one more level of fault tolerance, in that it holds the MFV to a needed position if the MFV does not produce output.

The MFV, BFV and FP controllers also send their outputs to the MC and BC. When the MC (or BC) is in control, it compares its output to the signals that the MFV, BFV and FP controllers output signal to the actuators. If the output signal differs, then the computer indicates to the MFV, BFV and FP controllers that it has failed. The digital feedwater controller failover logic consists of the following: the MC has control of the control points initially, with the BC in “hot” standby. If the MC fails, then the BC takes control. If the BC fails after the MC has failed, then the MFV, BFV, and FP controllers each use one of their recent output value from the computer (essentially the last one that the controller can store) and recycle that value to the control points. Any time a component fails, the operator console is notified to allow operators to take mitigating actions.

CHAPTER 3

DESCRIPTION OF THE SYSTEM OPERATION UNDER ABNORMAL CONDITIONS

This section presents a failure modes and effects analysis (FMEA) of the DFWCS [7]. The failure classes may include sensor failures, output failures, input failures and internal failures. Each of the failure classes may contain a large number of faults. For example, sensor failure may be the result of a physical sensor failure, cut wires, loose connections, or hardware (such as analog to digital converters) on the receiver failing. While these failure classes may be general, they are expected to capture the necessary information about possible failures of the benchmark feedwater control system. Each component type in the system, MC, BC, MFV controller, BFV controller, FP controller and PDI controller has a separate FMEA chart associated with that component. In addition, the actuated devices (i.e. MFV, BFV and FP) may fail to perform their design due to mechanical failure. The only mechanical failures that will be considered for the benchmark DFWCS are the valves getting stuck in their current position.

3.1 Main and Backup Computer FMEAs

Table 3.1 summarizes MC failure classes [7]. Sensor failures may occur from many possible sources, including decaying or broken wires, failed sensors, intermittent transmission failures, or analog to digital conversion errors. A sensor failure is detected through the use of rate of change checks, range checks, and comparison to previous values used by both the MC and BC. For illustration, if the sensor was reading a 1.5 feet water level signal and then it received a 150 feet signal, then this value is considered to be an invalid sensor reading. Also, an indicator light illuminates on the operator console.

The MC and BC each disregard an invalid sensor reading if there is one sensor of each type that is valid and wait for one computation interval before indicating their failure. Subsequently, a common mode sensor failure that causes one sensor of a type from both the MC and BC will not cause the MC and BC to fail. Rather, they will operate using only one sensor. Thus, the controlled process is not affected by a single sensor failure. However, due to the physical wiring of the sensors, if one sensor fails, its failure may affect sensor readings for several computers on both steam generators and may lead to a common mode failure. Even with this type of failure, the digital feedwater system can still maintain control. As in the case of single sensor failure described above, this event may occur from many possible sources including decayed or broken wires, failed sensors, and intermittent transmission failures. Multiple sensor failures are detected in the same manner as the single sensor failure described earlier.

The MC and BC uses previous values to perform their computations if multiple sensors fail as in the single sensor failure case. Also, the MC and BC notifies the MFV, BFV, and FP controllers that it considers itself failed if the sensor readings do not return to normal after a brief delay (which depends on the sampling requirements of the physical process). This notification forces the MFV, BFV, and FP controllers to switch their input acquisition device as necessary (i.e. they switch from the MC to BC, BC to MC). In case both the MC and BF are failed, all the controllers maintain the latest valid value. Power level changes are disabled in this mode. An indicator light illuminates on the operators' console.

Control could be affected if the MC and BC have invalid data and the MFV, BFV, and FP controllers are forced to control the process. The MC failure status signal can be activated if the MC takes itself down through sensor validity checks, through application failures or through a communication problem with the MFV, BFV, and FP controllers. The MC failure status signal can also be activated if the MC detects that the output it sent to the MFV, BFV or FP controllers differs from the output actually used by those controllers. Alternatively, the MC's watchdog timer may go off. Finally, the MC may fail and its failure may not be detected due to a communication failure with the MFV, BFV, and FP controllers. Failure is detected through the use of a watchdog timer and the computer's internal validity checks. If the watchdog timer goes off, it signals the MFV, BFV and FP controllers to notify them that the MC has failed. These are the only components that are affected by a MC failure.

The MC also checks and indicates that it is unreliable if any detectable errors occur. For instance, the MC indicates that it is unreliable if it can detect that its sensors readings are invalid. A detected failure causes the MC to signal failure and the system then transfers to using the BC. If a failure is not detected, then the system continues to use the

MC until either the output goes out of range or the rate of change exceeds what is allowed. Then the BC takes over. The impact of a detected failure is minimal as the BC takes control due to the watchdog timer reset. The system can maintain control. However, an undetected failure may act as a Byzantine[1] failure. For example, if the MC experiences a crash, possibly an arbitrary value may be the output to the BFV, MFV and FP controllers. This is one of the possible additional failure modes of digital I&C systems. Also, the MC's output may drift or simply fail to send an output signal. At some point, the output signal may change such that it goes out of range, the rate of change becomes too high or the output signal is lost. Should this situation occur, the MFV, BFV, and FP controller(s) detect the error and switch to the BC. However, there still may be a loss of control until the failure is detected. It is assumed that the arbitrary value category includes any Byzantine failures that may occur.

Failure type	**Detection of Failure**	**Effect of Failure on Controller**	**Effects on Controlled/Monitored Process Variables**
Loss of one sensor inputs of a type of input via computer diagnostics.	Computer detects loss in sensor reading.	Computer ignores failed sensor. If the sensor does not return (to valid input), computer indicates it has failed if the other computer is operating normally.	None. Backup Computer takes control if the sensor does not return.

Table 3.1 – FMEA for the MC

Continued

[1] A Byzantine failure is one in which any failed processes are modeled as actively trying to disrupt the normal goals of the system

Table 3.1 continued

Loss of both sensor inputs of a type of input via computer diagnostics.	Computer detects loss in sensor reading.	Computer uses the old values of the sensor reading. If the sensor reading does not return, computer indicates that it has failed.	None. At the worst case the Backup Computer takes control.
Intermittent sensor failure.	Computer detects out of range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return to valid return.	None. At the worst case the backup computer takes control.
Sensor failure.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensor does not return.
Both sensors fail.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensors do not return.
Loss of an output (0.0 vdc).	Component connected to computer detects 0.0 vdc input reading.	Component signals that this computer has failed.	None. The backup computer takes control.
Loss of Power.	Computer failed signal is tripped.	Continue with fail over logic.	None. The backup computer takes control and no effect on water level.
Roundoff/truncation/ sampling rate errors.	Detected by connecting components if output ever is out of range or exceeds the physically possible rate.	Component fails the computer.	If not detected, water level may drift. If detected, the backup computer takes control and no effect on water level.
Unable to meet needed response requirements.	Watchdog timer detects the failed computer.	Failover action occurs.	None. Backup computer takes control with no effect on water level.

Continued

Table 3.1 continued

Watchdog timer fails to activate.	Detectable when outputs of computer go out of range or exceed the physically possible rate.	Failover action occurs if detected.	If not detected, water level may drift. If detected, the backup computer takes control with no effect on water level.
Watchdog timer activates when computer has not failed.	Not detectable.	Failover action occurs.	None. Backup computer takes control.
Arbitrary value output.	Detectable by connected component if the computer does not reset the watchdog timer.	Component connected to output signals that this computer has failed.	If not detected, water level may increase or decrease. If detected, the backup computer takes control and no effect on water level.
MFV/BFV/FP controllers do not use the output that the MC computed.	Detected by comparing outputs of MFV/BFV/FP controllers with MC output	Component initiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Setpoint drift.	Detectable by the computers once the setpoint has drifted out of range.	Computers revert to using a preprogrammed value.	Water level will increase or decrease until it reaches the setpoint that has drifted out of range, then water level will settle to the preprogrammed value.

Table 3.2 shows BC failure modes [7]. A simultaneous MC and BC failure is detected through the use of watchdog timers. If the watchdog timers go off, the MFV, BFV and FP controllers are notified of the computer failures. The MFV, BFV, and FP controllers are

the only components that are affected directly by this failure. An indicator light illuminates on the operators' console. The failure is mitigated by the action of the MFV, BFV, and FP controllers to hold the outputs at their old values.

The impact of these failures is, as expected, quite significant. In this case, the digital control system has failed and the operators must intervene to take manual control of the process as the MFV, BFV, and FP controllers continue to output old values to the MFRV, BFRV, and FP controllers. The problems resulting from a MC and BC failure get worse if the failures are not detected, as the MFV, BFV, and FP controllers will take more time before they take over.

Failure type	**Detection of Failure**	**Effect of Failure on Controller**	**Effects on Controlled/Monitored Process Variables**
Loss of one sensor inputs of a type of input via computer diagnostics.	Computer detects loss in sensor reading.	Computer ignores failed sensor. If the sensor does not return (to valid input), computer indicates it has failed.	None.
Loss of both sensor inputs of a type of input via computer diagnostics.	Computer detects loss in sensor reading.	Computer uses the old values of the sensor reading. If the sensor reading does not return, computer indicates that it has failed.	The valves and feedwater pump will remain in the same state if the failover to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.

Table 3.2 – FMEA for the BC

Continued

Table 3.2 continued

Intermittent sensor failure.	Computer detects out of range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return to valid input.	The valves and feedwater pump will remain in the same state if fail over to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Sensor failure.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensor does not return.
Both sensors fail.	Main and backup computers detect this via range output and physically impossible rates of change.	Computer ignores sensor input and uses old values. It fails itself if sensor does not return.	Both the main and backup computers will fail themselves if the sensors do not return.
Loss of an output (0.0 vdc).	Component connected to computer detects 0.0 vdc input reading.	Component signals that this computer has failed.	None. The backup computer takes control.
Loss of Power.	Computer failed signal is tripped.	Continue with fail over logic.	The valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.

Continued

Table 3.2 continued

Roundoff/truncation/ sampling rate errors.	Detected by connecting components if output ever is out of range or exceeds the physically possible rate.	Component fails the computer.	If not detected, unknown. If detected, the valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration.
Unable to meet needed response requirements.	Watchdog timer detects the failed computer.	Failover action occurs.	The valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Watchdog timer fails to activate.	Detectable when outputs of computer go out of range or exceed the physically possible rate.	Failover action occurs if detected.	If not detected, unknown. If detected, the valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration.

Continued

Table 3.2 continued

Watchdog timer activates when computer has not failed.	Not detectable.	Failover action occurs.	The valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Arbitrary value output.	Detectable by connected component if the computer does not reset the watchdog timer.	Component connected to output signals that this computer has failed.	If not detected, unknown. If detected, the valves and feedwater pump will remain in the same state if fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
MFV/BFV/FP controllers do not use the output that the MC computed.	Detected by comparing outputs of MFV/BFV/FP controllers with MC output	Component initiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Setpoint drift.	Detectable by the computers once the setpoint has drifted out of range.	Computers revert to using a preprogrammed value.	Water level will increase or decrease until it reaches the setpoint that has drifted out of range, then water level will settle to the preprogrammed value.

3.2 MFV, FP, BFV and PDI controllers

The FMEA for MFV, BFV and FP controllers are given in Tables 3.3, 3.4, and 3.5, respectively [7]. The MFV controller may fail due to a power loss or an internal program crash (possibly from hanging, hard crash, or output failure). This event can only be detected if the MFV controller output drops to 0.0 volts. In this case, the PDI controller senses the drop in output and asserts the old value of the MFV controller's output to the MFV. The PDI controller acts as the MFV would, outputting the old output to the MFV. At best, the digital feedwater control system reverts to a manual mode (and fails). At worst, the MFV controller does not output the correct signals to the MFV and can cause loss of control of the process.

The BFV controllers may fail because of a power loss or a program crash. This event cannot be detected by the digital feedwater control system. If the BFV controller does not output the correct signals to the BFV, loss of control of the process may occur. The MFV/BFV/FP controllers may disagree as to the failure states of each computer. It is then possible for the MFV controller to be accepting different signals than the FP and BFV controllers. This event cannot be detected. The impact of this failure is that the SG water level may increase or decrease depending on the failure of the MC/BC and the MFV/BFV/FP controller's indication of those failures.

Also, communication between the MFV, BFV, FP and PDI controller occurs only to coordinate failure detection. This communication type of failure is captured by the computer erroneously reported failed or not failed. Finally, it is noted that the MFV, BFV, PDI and FP controllers are all digital also, and as such may experience crashes that may cause them to be unable to detect failures or output arbitrary values.

Failure type	**Detection of Failure**	**Effect of Failure on Controller**	**Effects on Controlled/Monitored Process Variables**
Loss of power.	Detected by PDI.	PDI uses old signal for the MFV.	The MFV will remain in the same state. Thus the effect on the process variables depends on the event in consideration.
Loss of output signal.	Detected by PDI.	PDI uses old signal for the MFV.	The MFV will remain in the same state. Thus the effect on the process variables depends on the event in consideration.
Computer erroneously reported failed.	Not detected.	Component initiates failover operation.	The MFV will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Computer erroneously reported not failed.	Can be detected if computer fails and output of computer is out of range, the rate of change is too great, or the watchdog timer goes off.	Component initiates failover operation.	If detected the MFV will remain in the same state if the fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration. Otherwise, unknown.

Table 3.3 – FMEA for the MFV

Continued

Table 3.3 continued

MFV, BFV, FP controllers do not agree from which computer to accept input.	Cannot be detected.	MFV, BFV, and FP may receive inconsistent input.	Unknown effect may increase or decrease water level.
High output.	Cannot be detected.	MFV will be driven open.	Feedwater flow will increase, causing the water level to increase.
Low output.	Cannot be detected.	MFV will be driven shut.	Feedwater flow will decrease unless the computer is operating in low power mode. In low power mode, there will be no effect.
Arbitrary value output.	Cannot be detected.	MFV will open/close according to the output.	Unknown effect; may increase or decrease water level.

Table 3.4 shows the FMEA table for the BFV controller.

Failure type	**Detection of Failure**	**Effect of Failure on Controller**	**Effects on Controlled/Monitored Process Variables**
Loss of power.	Not detected.	Nothing.	Unknown effect on water level.
Loss of output signal.	Not detected.	Nothing.	Unknown effect on water level.

Table 3.4 – FMEA for the BFV

Continued

Table 3.4 continued

Computer erroneously reported failed.	Not detected.	Component initiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.
Computer erroneously reported not failed.	Can be detected if computer fails and output of computer is out of range, the rate of change is too great, or the watchdog timer goes off.	Component initiates failover if detected.	If detected the valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration. Otherwise, unknown.
MFV, BFV, FP controllers do not agree from which computer to accept input.	Cannot be detected.	MFV, BFV, and FP may receive inconsistent input.	Unknown effect may increase or decrease water level.
High output.	Cannot be detected.	BFV will be driven open.	Feedwater flow will increase, causing the water level to increase.
Low output.	Cannot be detected.	BFV will be driven shut.	Feedwater flow will decrease unless the computer is operating in low power mode. In high power mode, there will be no effect.
Arbitrary value output.	Cannot be detected.	BFV will open/close according to the output.	Unknown effect; may increase or decrease water level

The FP controller may fail because of a power loss or a program crash as the MFV or BFV controller could. This event cannot be detected by the DFWCS. The FP controller would not output the correct signal to the feedwater pump and thus would be unable to control the feedwater pump.

Failure type	**Detection of Failure**	**Effect of Failure on Controller**	**Effects on Controlled/Monitored Process Variables**
Loss of power.	Not detected.	Nothing.	Unknown effect on water level.
Loss of output signal.	Not detected.	Nothing.	Feedwater flow will decrease to the minimal value, possibly decreasing the water level.
Computer erroneously reported failed.	Not detected.	Component initiates failover operation.	The valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers. Thus the effect on the process variables depends on the event in consideration.

Table 3.5 – FMEA for the FP

Continued

Table 3.5 Continued

Computer erroneously reported not failed.	Can be detected if computer fails and output of computer is out of range, the rate of change is too great, or the watchdog timer goes off.	Component initiates failover if detected.	If detected the valves and feedwater pump will remain in the same state if the fail over fails to the MFV/BFV/FP controllers, thus the effect on the process variables depends on the event in consideration. Otherwise, unknown.
High output.	Cannot be detected.	FP will be driven to increased pressure.	Feedwater flow will increase, causing the water level to increase.
Low output.	Cannot be detected.	FP will be driven to decreased pressure.	Feedwater flow will decrease, causing the water level to decrease.
Arbitrary value output.	Cannot be detected.	FP will speed up/slow down according to the output.	Unknown effect. May increase or decrease water level.
MFV, BFV, FP controllers do not agree from which computer to accept input.	Cannot be detected.	MFV, BFV, and FP may receive inconsistent input.	Unknown effect. May increase or decrease water level.

The FMEA table for the PDI controller is shown in [7]. The PDI controller may also fail because of a power loss or a program crash as the MFV, BFV and FP controllers and event will not be detected by the digital feedwater control system. If the PDI controller fails by loss of output, then a failed MFV controller will not be detected. If the PDI fails by sending a spurious signal to the MFV when the MFV controller has not failed, then the MFV and PDI controller outputs will sum and give an increased signal to the MFV, causing the MFV to open more than it should.

Failure type	**Detection of Failure**	**Effect of Failure on Controller**	**Effects on Controlled/Monitored Process Variables**
Loss of inputs (0.0 vdc).	PDI detects a 0.0 vdc input.	PDI outputs an old value of the MFV controller output to the MFV.	The MFV valve will receive a signal that is the sum of the PDI and MFV controller signals, thus increasing feedwater flow.
Loss of power.	Not detected.	None.	None directly, unless the MFV controller has failed, then resulting effect is unknown.
Loss of Outputs.	Not detected.	None.	None directly, unless the MFV controller has failed, then resulting effect is unknown.
Arbitrary failure.	Not detected.	PDI can output any value to MFV.	The MFV valve will receive a signal that is the sum of the PDI and MFV controller signals, thus increasing feedwater flow.

Table 3.6 – FMEA for the PDI

CHAPTER 4

FINITE STATE MACHINE MODELING

As presented in Section 1 and Fig.1.1, the next step in the modeling of digital control systems using the Markov/CCMT after the definition of the FMEA is the construction of a finite state machine model for each component of the DFWCS [7]. The purpose of this section is to show how it is possible to construct a finite state machine model from the FMEA presented in Section 3.

4.1 From the network structure to the system finite state machine

As a general discussion, the design of a system finite state machine for digital control systems takes into account several aspects that can be summarized as follows:

1. *The structure and the topology of the network.* The concept of topology applied to networks refers to how data are shared and passed among the components connected to the network itself. Figure 4.1 shows the basic connection schemes which it is possible to encounter in the analysis of digital control systems. Figure

4.1 also indicates how the DFWCS described in Section 2 can be considered as a simple mesh scheme.

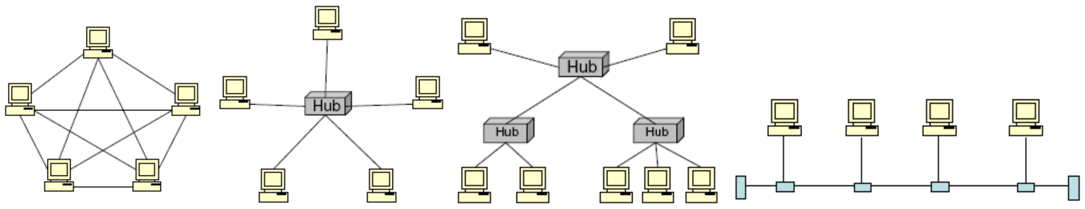


Fig.4.1 - Possible connection scheme: mesh (e.g., DFWCS), star, tree, bus

2. *The format of the data*. Data on the line can be transmitted in two different formats: analog or digital. From [10] an analog signal is defined as a continuous wave form that changes smoothly over time; as the wave moves from the minimum to the maximum value it passes through and includes an infinite number of values along its path. A digital signal, however, has only a limited number of defined values depending on the encoding scheme. From a reliability view point the format of the data affects the resilience property of the communication systems from external events and how the communication system itself is able to detect and correct errors in the data transmitted on the line.

3. *Transmission media*. Different types of media can be used to transmit data. Guided (e.g., coaxial or fiber-optic cable) and unguided (e.g., electromagnetic waves used without a physical conductor) media can be used.

In building a finite state machine for each component (computers, controller, etc.) of the DFWCS, states and transition must reflect the FMEA presented in Section 3. Redundant components are modeled together by merging the finite state machines of the redundant components into a single finite state machine. Moreover, the inter-component dependability (for example, due to the topology of the network) is taken into account inserting the finite state machine of a component inside specific states of another component.

The DFWCS can be regarded as consisting of three layers of interaction [7]:

- Intra-computer interactions: a layer which describes the status of the single computer (MC or BC)
- Inter-computer interactions: a layer which describes the status of the set of both computers (MC and BC)
- Computer-controller-actuated device interactions: a layer which describes the status of the controllers (MFV, BFV and FP controllers).

In Sections 4.2, 4.3 and 4.4 these three layers are presented and described.

4.2 Intra-computer interactions

The intra-computer interactions layer consists of 5 states (see Fig.4.2) [7]. These interactions can be regarded as transitions between the possible states of a single computer. In State A, the computer is operating correctly. In State B, the computer detects loss/invalid output for 1 sensor of any type (e.g. water level). State C represents loss/invalid output for 2 sensors of any one type. In state D the computer has detected an internal problem and is signaling that it has to be ignored. In State E, either the sensor output is invalid or there is an internal processing error in the computer; however, the computer does not detect the fault and is transmitting the wrong information to the controllers. These states capture the possible failures in the FMEAs presented in Section 3.2, and that occur within both the MC and BC.

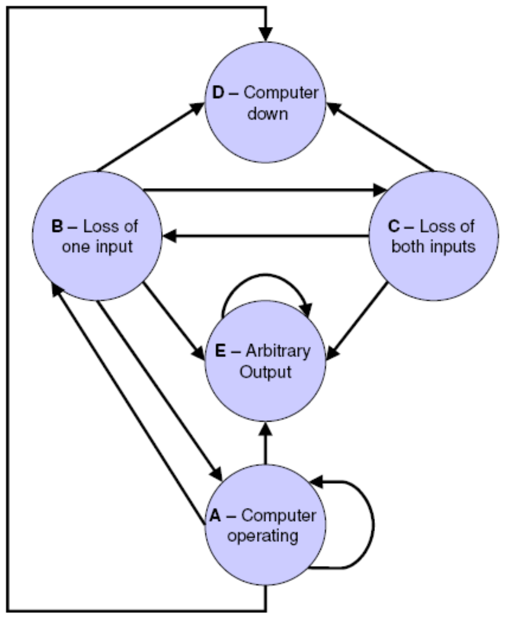


Fig.4.2 – Intra-computers interaction scheme

4.3 Inter-computer interaction

The inter-computer interaction layer displayed in Fig.4.3 [7], shows the interactions between the two computers (MC and BC). In particular, the transfer of control from the MC to the BC is represented here.

In this layer, 3 computer macro-states (MSs) are identified. Each of these macro-states indicates the status of both the computers:

1. In State 1 both MC and BC are operating normally.
2. In State 2, one computer is operating correctly and the other is down but can be recovered.
3. In State 3, again one computer is operating correctly and the other is down but it is not recoverable.

Transitions between the MSs depend upon the state of the controlling computer. Primary and secondary computers correspond, respectively, to the computer that is sending data to the controller and to the computer that is waiting in hot standby. Both the MC and BC can be the primary or the secondary computer.

Transitions from MS1 to MS2 are due to recoverable failures while transitions from MS1 to MS3 are due to not recoverable failures. Recoverable and non-recoverable failures are defined as the following:

- Recoverable failure corresponds to the momentarily inability for the computer (which is still operating correctly) to send valid data to the controller (e.g. due to a loss of input from one couple of sensors). Since, it is possible to recover from type of this failure, transitions from MS2 to MS1 are possible.

- Non-recoverable failure corresponds to an internal failure of the computer (e.g. the trip of the watchdog timer) or to a loss of output of the computer itself. Since, it is not possible to recover this failure: transitions from MS3 to MS1 are possible.

4.3.1 Transitions from MS1 to MS2

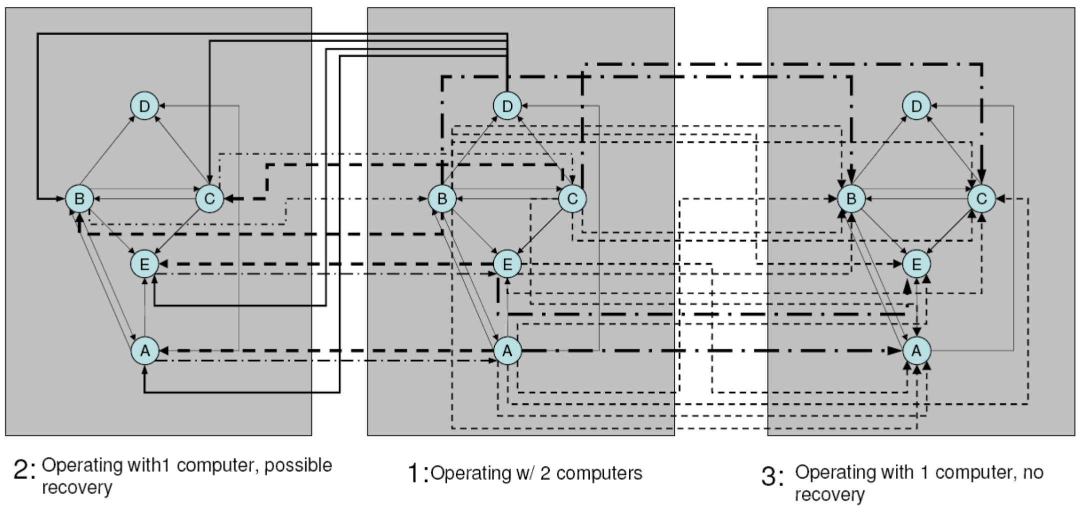
If the secondary computer (i.e. the computer that is not in control) fails and it is still recoverable, a transition from MS 1 to MS 2 occurs (see Fig.4.3). These transitions simply take each state in MS 1 to the corresponding state in MS 2. For example, State A (or operational) in MS 1 would have a transition to State A in MS 2. When the secondary computer recovers, the opposite transitions occur (from MS 2 to MS 1). Again, for example, if the operating computer is in State A, MS 1, then the transition would start there and end in State A in MS 1.

From the State D in MS 1 the possible transitions represent the takeover of control of the process by the secondary computer (which from now on will be regarded as the primary computer). The transitions from this state go to all states except for State D in MS 2. The rationale behind these transitions is that the secondary computer was operating and may have transitioned to states other than State A in MS 2. The reason that State D in MS 2 is not a possible destination is that if State D is reached by the secondary computer, then another transition from MS 1 to MS 2 must have already taken that into account.

4.3.2 Transitions from MS1 to MS3

The fail-over action from MS 1 to MS 3 is a result of controller action via the watchdog timer or detecting the output failure from the computer. This action takes down the failed computer permanently and can occur to both the primary and secondary computer. If it occurs to the secondary, the transitions mimic the action of the secondary failure transitions from MS 1 to MS 3 by simply transitioning from a state in MS 1 to the respective state in MS 3. For example, State A in MS 1 would have a transition to State A in MS 3.

If the primary computer fails in a non-recoverable manner when both MC and BC are operating (i.e. when the DFWCS is in MS1), then the DFWCS can go to any state in MS 3 except State D by the same rationale for transitions between MS1 and MS2. The transitions must take into account that the secondary computer may have already entered different states and these must be represented in the transitions to MS 3.


D
B
C
E
A
2: Operating with1 computer, possible recovery
1: Operating w/ 2 computers
3: Operating with 1 computer, no recovery

Computer States
A: Operating
B: Loss of one input
C: Loss of both inputs
D: Computer down
E: Arbitrary output

Macro States
1: Controller is receiving data from both computers
2: Controller is receiving data from 1 computer while the other one can be recovered
3: Controller is receiving data from 1 computer while the other one can not be recovered
Freeze: Controller sends the same data to the valves from the previous time step

▬ ▬ ▬ ·	Secondary goes down (recoverable)	────	Primary release control of the process.	▬ · ▬ ·	Secondary computer watchdog timer trips or loss of output to controller	- - - - - - -	Common cause sensor failure
- · - · - · -	Secondary recovers	- - - - - - -	Primary computer watchdog timer trips or loss of output to controller.	▬▬ ▬▬ ·	Primary goes down. Secondary unavailable		

Fig.4.3 – Inter-Computers interaction scheme

4.4 Computer-controller-actuated device interactions

Figure 4.4 shows all the possible controller-computer-actuated device interactions [7] according to the FMEA charts presented in Section 3.3. The shaded circles represent signals to the actuated devices (MFV, BFV, FP) upon computer/controller failure, as well as the mechanical failure of the actuated device (Device Stuck). As indicated in the beginning of Section 2, mechanical failure of the actuated device leads to the device maintaining its current position for MFV and BFV or to zero flow for FP.

The planes represent the communication status between the controller and actuated devices. The two-way transitions between Planes I and II are necessary to keep track of the computer from which the controller is receiving data when the communications between controller and actuated device are restored.

The scheme shown in Fig.4.4 shows the connection between a single controller (e.g., the MFV controller) and the computers (MC and the BC) and its own actuated device (MFV). In particular, the following types of controller failures are under consideration:

- *Arbitrary output:* random data are generated and sent to the actuated device (i.e., MFV, BFV and FP)
- *Output high*: output value is stuck at the maximum value (i.e. valve totally open or pump at the maximum speed)
- *Output low*: output value is stuck at the minimum value (i.e. valve totally closed or pump stopped)
- *0 vdc output*: loss of communications between controller and actuated device
- mechanical failure of the actuated device

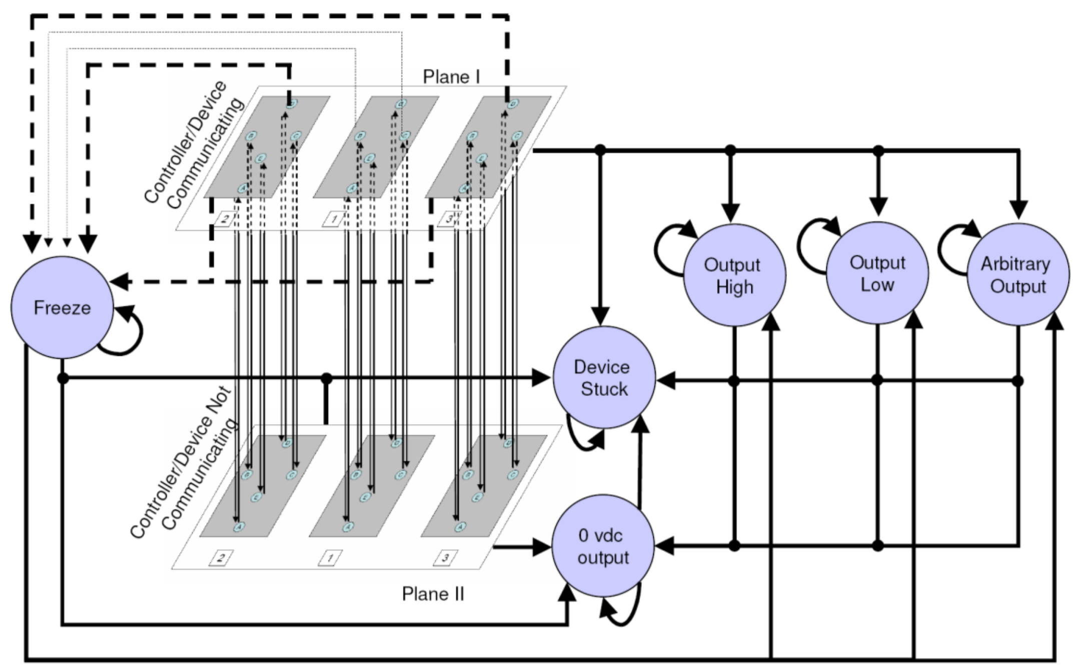


Fig.4.4 – Computer-Controller-Actuated device interactions scheme

Moreover, as a result of the failure of both computers, the controller can recognize the failure and send to the actuated devices (i.e. pump or valves) the old valid value (i.e. Freeze). If the controller does not recognize the failure, then it will simply pass on false information (Arbitrary Output) to the actuated device. Figure 4.4 also shows how the computer-computer interactions (presented in Fig.4.3) integrate with computer-controller and controller-actuated device interactions.

Device Stuck refers to mechanical failures and is independent of the failure modes of the computers and controllers. The behavior of the controller under normal and failed operation can be described as follows:

- When both MC and BC are down, the controller transits to the Freeze state. The actuated device remains in the position corresponding to the last valid value.
- If the controller is operating and an Output High or an Output Low or an Arbitrary Output failure occurs, the controller transits to the corresponding state and the actuated device assumes the highest, the lowest or an arbitrary position, respectively.
- If the controller is in the Freeze state and an Output High, Output Low or Arbitrary Output failure occurs, the controller transits to the corresponding state and the actuated device assumes the highest, the lowest or an arbitrary position, respectively.
- If a loss of output occurs when the controller is failed (i.e. the controller is sending Arbitrary Output, Output High or Output Low state), then the actuated device receives a 0 vdc as input which correspond to the lowest position.

4.5 Examples of Initiating Events

In this section two scenarios are presented in order to show how the finite state machine description (see Fig.4.2, Fig.4.3, and Fig.4.4) described above can be used to model several accident scenarios which have taken place in existing nuclear power plants.

4.5.1 Scenario 1

The scenario starts with the benchmark system operating at high power at 90% reactor power with the MC controlling the MFV, BFV, and FP, the BC operating correctly and the MFV, BFV, FP, and PDI controllers operating correctly (i.e., state A of MS1 of Fig.4.5). When the FP controller was installed, the MC to FP and BC to FP failure signal wire were not soldered correctly [1]. As a result of the improper soldering and vibration within the plant, the integrity of these connections has become compromised. Additionally, as has been encountered in operating nuclear power plants [12], corrosion problems have affected the wiring.

In this scenario, the corrosion affects one of the water level sensor wires from level sensor 1 (LVL1). As a consequence of the corrosion, the MC receives intermittently no signal from LVL1 over the course of several months. Due to wear-out affecting the connection from level sensor 2 (LVL2), the sensor is unable to transmit a signal and fails to report a value (resulting in 0.0 DC volts on the line). The MC senses this signal loss and ignores the invalid input.

At this point, corrosion in the connection from LVL1 described above causes the connection to fail completely, resulting in 0.0 DC volts on the line. Consequently, no signal is received by the MC from LVL1. At this point, all level sensor transmissions to the MC have failed. The MC waits for one processing cycle, determines the level sensor inputs are still unavailable and then signals failure to the MFV, BFV, and FP controllers. As a result of the MC failure, the MFV, BFV, and FP controllers each attempt to transfer control to the BC (i.e., looking at Fig.4.5, a transition from state A to state D of MS1 occurs and then a transition from state D of MS1 to state A of MS2 occurs).

Due to vibration, the FP controller's MC and BC failure status wires become completely disconnected from the controller. This disconnection causes the FP controller to consider both the MC and BC as failed. By following its fail over procedure, the FP controller recycles its last good output. The FP controller then signals MC and BC failure to the MFV and BFV controllers. This causes the MFV and BFV controllers to recycle their last good outputs until operators intervene. (i.e., looking at Fig.4.5, a transition from state A of MS2 to state D of MS2 occurs and then a transition from state D of MS2 to state Freeze of MS2 occurs).

1	Level sensor two fails due to a broken connection.
2	Level sensor one fails as the corrosion causes the wire to degrade too much to be usable.
3	MC activates failure status signal.
4	MFV, BFV, and FP controllers transfer control to the BC.
5	MC to FP and BC to FP controller failure status lines become disconnected due to vibration.
6	FP signals BC and MC failure to other controllers and uses last good output value.
7	MFV and BFV controllers use last good output

Table 4.1 – List of events for the first scenario

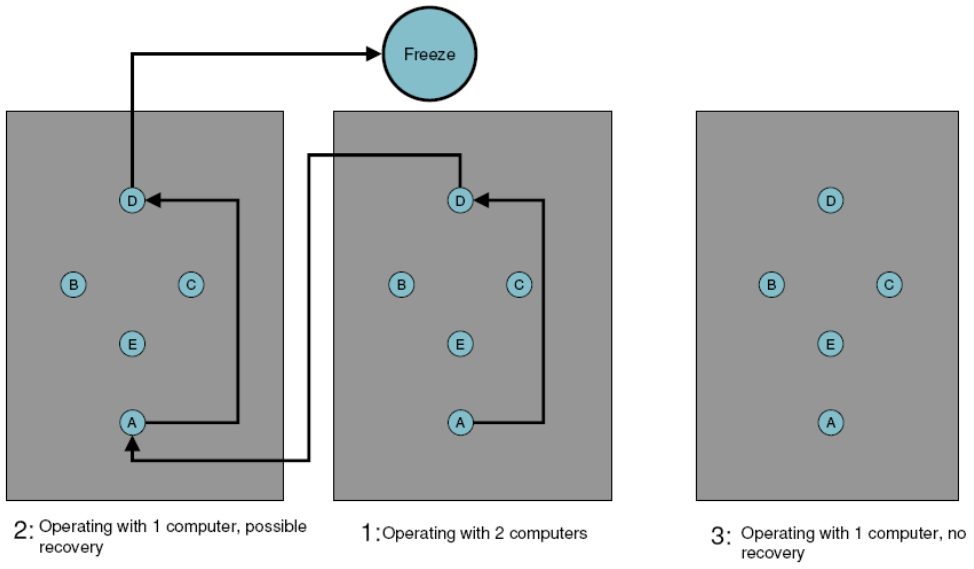

Freeze
D
B
C
E
A
2: Operating with 1 computer, possible recovery
1: Operating with 2 computers
3: Operating with 1 computer, no recovery

Fig.4.5 – Description of the first scenario through the finite state machine

4.5.2 Scenario 2

Scenario 2 starts with the system operating normally in high power at 90% reactor power with the MC in control and all other controllers and components operating normally (i.e., state A of MS1 of Fig.4.6). The BC fails and causes the watchdog timer to expire. This timer expiration is detected as BC failure by the MFV, BFV, and FP controllers (i.e., looking at Fig.4.6, a transition from state A of MS1 to state A of MS3 occurs).

Due to corrosion of an inline power supply fuse, the power supply to the MC shuts down [11, 12], resulting in an MC failure (i.e., looking at Fig.4.6, a transition from state A of MS3 to state D of MS3 occurs). The MFV, BFV, and FP controllers detect the loss of the MC and take control holding the valves and pumps settings at their current values until operators are able to intervene (i.e., looking at Fig.4.6, a transition from state D of MS3 to Freeze state occurs). Scenario 2 is summarized in Table III. This scenario demonstrates the benchmark system's ability to fulfill requirements 2 and 9, namely the representation of electrical power needs and the use of the watchdog timer.

1	BC's watchdog timer expires. MFV, BFV, and FP controllers take control and maintain pumps/valve settings at current values until operators intervene.
2	MC fails: MC power supply fails.
3	MFV, BFV, and FP controllers take control and maintain pumps/valve settings at current values until operators intervene.

Table 4.2 – List of events for the second scenario

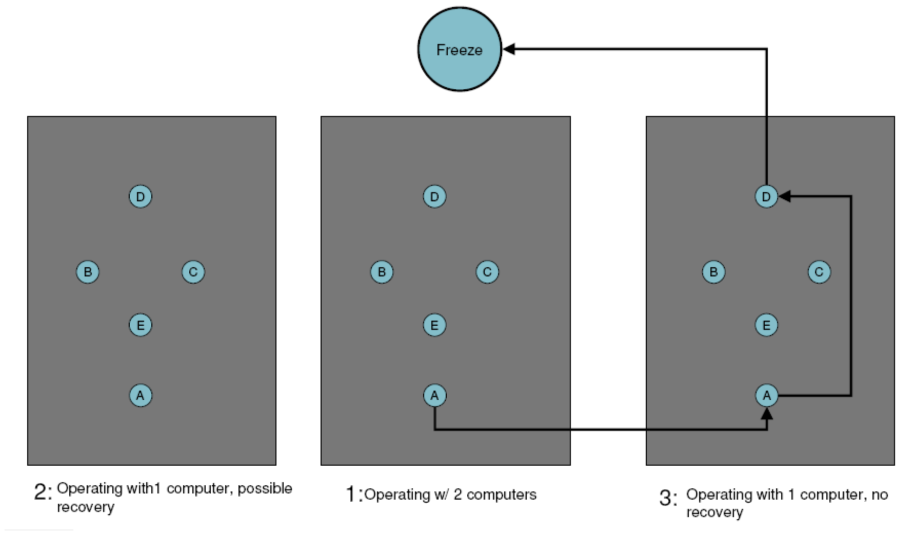

Freeze
D
B
C
E
A
D
B
C
E
A
D
B
C
E
A
2: Operating with1 computer, possible recovery
1 :Operating w/ 2 computers
3: Operating with 1 computer, no recovery

Fig.4.6 – Description of the first scenario through the finite state machine

CHAPTER 5

SIMULINK MODEL OF THE DFWCS

In order to perform the tuning and to confirm the validity of the control laws deduced from the control algorithm of the DFWCS (written in C code) of an operating PWR, a Simulink model has been built. This model is divided into modules in order to keep the model easy to tune, adaptable and comprehensible. The main modules are the following:

- Input module and SG module
- Control logic module
- Actuated device module.

In the following paragraphs, each of these modules is described in detail.

5.1 Input and SG modules

The input module includes a set of inputs (see Fig.5.1) which represent the power generated in the primary circuit. In this model it is supposed that the steam flow exiting the SG is proportional to the power generated by the reactor. A detailed model of the SG is not included and thus, the level of the SG is determined by a simple mass balance equation (see Eq.2.1).

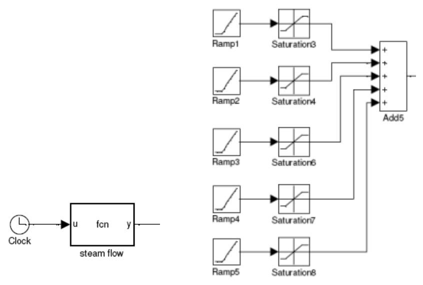
In particular, two kind of power profile has been analyzed:

1. Power decreasing exponentially due to the reactor shutdown (see Fig.5.2-a). In this case, the power *P(t)* of the reactor, previously operating at full power (3000 MW_{th}) for more than one year ($3.15\ 10^7$ s), drops almost instantaneously from 100% to 6.6% and then it starts to decrease exponentially following Eq.5.1 [13]:

$$P(t) = 0.066 \cdot P(0) \cdot \left[\frac{1}{(10+t)^{0.2}} - \frac{1}{(3.15 \cdot 10^7 + t)^{0.2}} \right] \qquad (5.1)$$

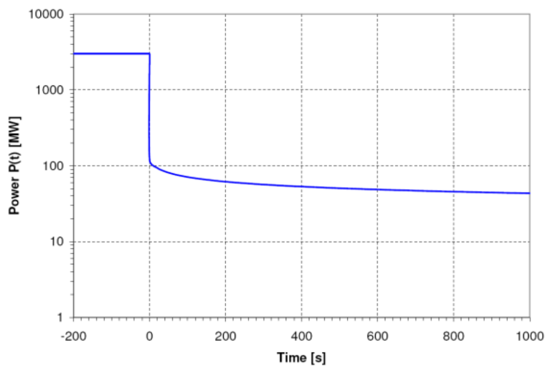
2. Power increasing from 70% to 72% through a sequence of brief ramps (see Fig.5.2-b).

Figure 5.1 displays how these input models have been implemented. In particular Fig.5.1.a shows how the power decay after a shutdown is implemented through a separate math file (.m file) while Fig.5.1.b shows the implementation of the sequence of ramps.

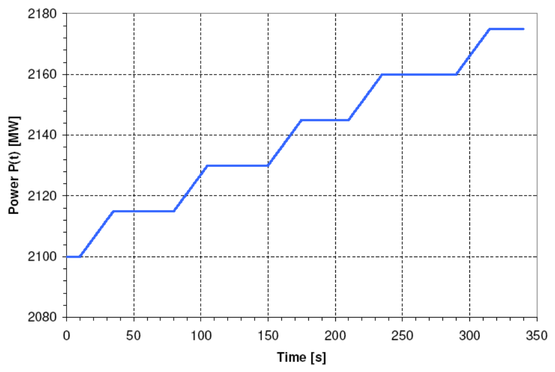


(a) (b)

Fig.5.1 – Simulink Input modules: exponentially decay (a) and sequence of ramps (b)



(a)



(b)

Fig.5.2 – Power behavior for the two cases under consideration: exponentially decay (a) and sequence of ramps (b)

5.2 Control logic module

The control logic module, shown in Fig.5.3, implements the control laws presented in Section 2 for both the High and Low power mode. Transition from High to Low power and vice versa are also realized. In this module, Eqs.2.2-2.6 are implemented.

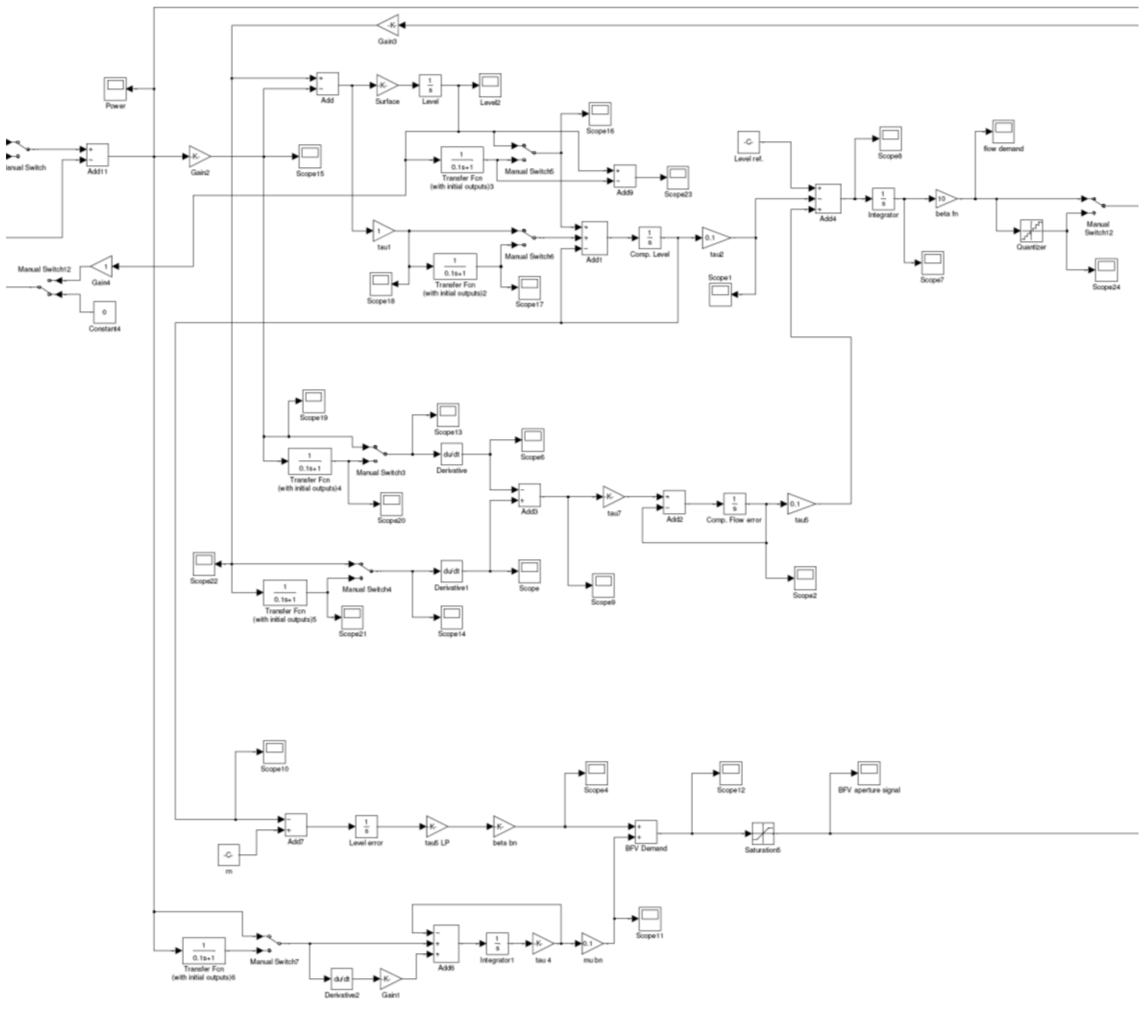


Fig.5.3 – Control logic module

5.3 Actuated device module

This module implements the response of the actuated devices (MFV, BFV and FP). In particular it translates the signal representing the feedwater demand coming from the control logic module into position of the MFV and BFV and speed of the FP. Here are implemented the lookup tables presented in Section 2 and shown in Fig.2.6. Mechanical (i.e., failed totally open or totally closed) and control failures (i.e., random data send to the actuated device or old valid value maintained in output) are also implemented here using time dependent blocks.

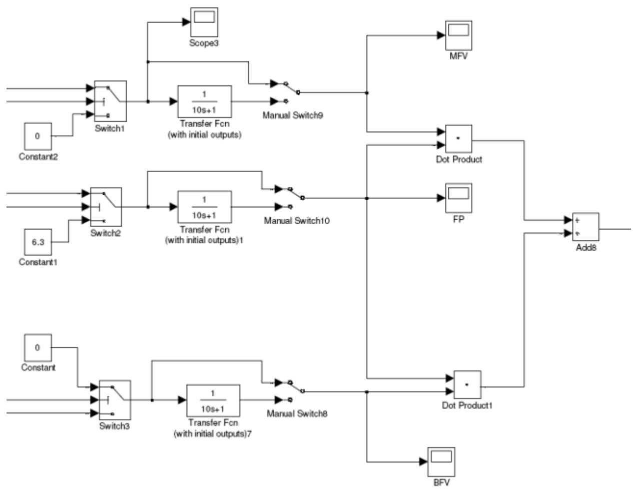


Fig.5.4 – Simulink module for MFV, BFV and FP

5.4 Initiating Event Simulation Results

As presented in Section 2, the equations representing the control laws of the DFWCS have been deduced from the C-code the DWCS controller of an existing PWR. It was not possible to deduce all the internal variables contained in these equations and the controller needed to be tuned.

The purpose of this section is to show the response of the digital feedwater model under the scenarios presented in Section 5.1 and Fig.5.2. The main concern is to obtain a stable response of the model under several power excursions. Moreover, the responses of the three actuated devices (i.e., MFV, BFV and FP) have to be reasonable in terms of rate of change of position (for MFV and BFV) or rate of change of speed (for FP). Sections 5.4.1 and 5.4.2 show the results obtained for the two scenarios presented in Section 5.1.

5.4.1 Reactor Shutdown

As presented in Section 5.1 the power is decaying exponentially starting from a value of 6.6% in this scenario. Since this scenario takes place in Low Power mode, the FP speed is set at 63% and only the BFV affects the feedwater flow (see Section 2). Figure 5.5 shows how the level of the SG is stabilizing during this scenario.

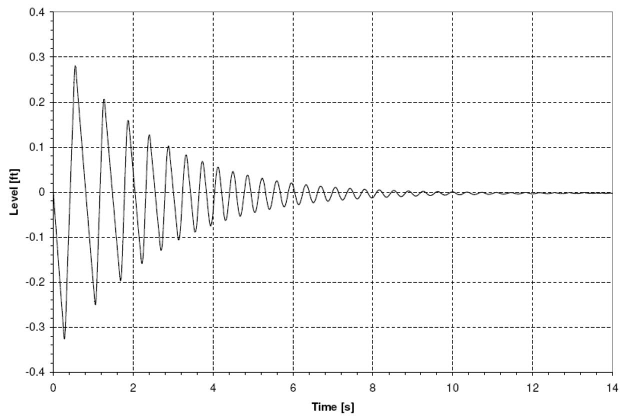
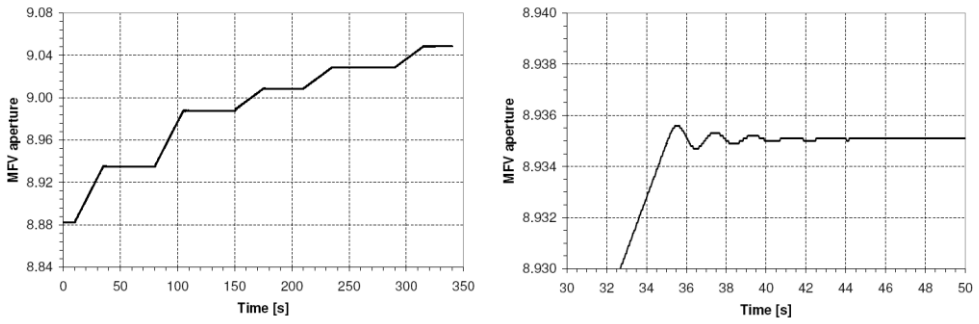


Fig.5.5 – Level response during the decay heat scenario

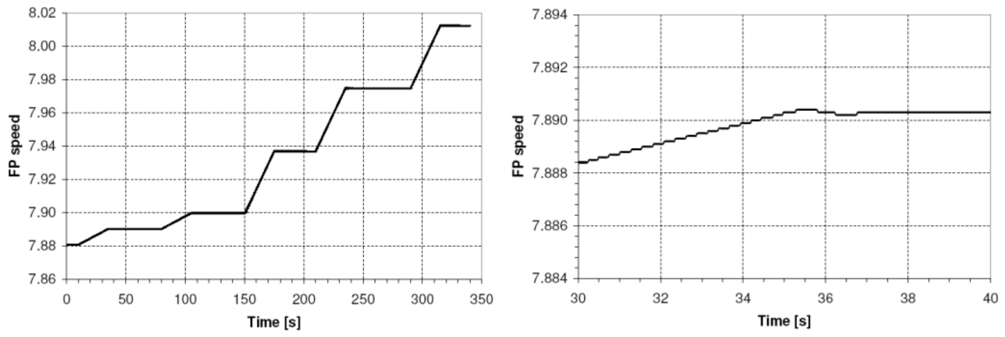
5.4.2 Power Increase Scenario

Figures 5.6 and 5.7 show the response of the FP and MFV for the second scenario presented in Sec.5.1 and Fig.5.2.b. For both MFV and FP the scale is ranging form 1 to 10.



(a) (b)

Fig.5.6 – MFV response for the scenario presented in Fig.5.2.b



(a) (b)

Fig.5.7 – FP response for the scenario presented in Fig.5.2.b

The rate of change of the power generated by the reactor strongly affects the oscillations of both MFV and FP[2]. Moreover, since each ramp performs a 0.5% increment

[2] Since the power is always above 15% the BFV is never used for the entire scenario.

of power (this value agrees with the normal operation of a typical PWR during a power excursion when control rods are moved out of the core by small amounts), the FP and MFV response for this scenario are very smooth as shown in Fig.5.6 and Fig.5.7. The main purpose of the DFWCS is to maintain the level of the SG between -24 and 30 inches (-2 and 3 ft) and Fig.5.8 shows that the behavior of the level of the SG for the scenario presented in Fig.5.2.b is consistent with this purpose.

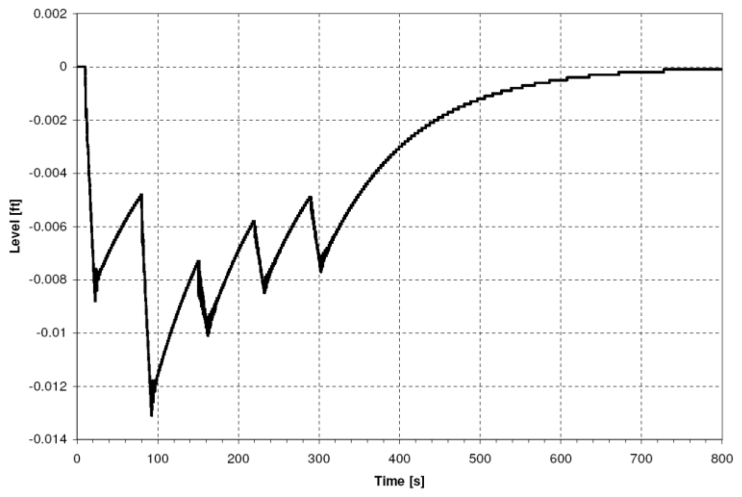


Fig.5.8 – Behavior of the level of the SG for the scenario presented in Fig.5.2.b

CHAPTER 6

MARKOV MODELING

In general, the construction of a Markov transition [14, 15] diagram for a single component assumes that:

- a set of mutually exclusive and exhaustive states n_m ($m=1,...,M$; $n_m =1,...,N_m$) has already been defined for component m, i.e.,

$$\sum_{m=1}^{M} p(n_m) = 1$$
$$\forall n_m, n_{m'} \rightarrow n_m \cap n_{m'} = 0 \tag{6.1}$$

- probability of transitions between states has been determined.

The choice of the failure states is based on the FMEA tables presented in Section 3. The Markov modeling of each component of the DFWCS (see Section 2.1) is presented in Sections 6.1 through 6.4. Section 6.5 shows how the grouping of several components into macro-components can be useful to simplify the modeling and to reduce the computational effort. Section 6.4 presents some considerations regarding possible rearrangement of the states of the components in order to create Markov transition diagrams that are independent from each other.

6.1 Main and Backup Computers

In Section 4 it is shown how it is possible to obtain the finite state machine model for both the MC and BC given the FMEA presented in and . In this model, sensors have been incorporated in the computer Markov transition diagrams. This merging procedure has been widely used in the Markov/CCMT methodology to minimize the overall number of states of the DFWCS (components state combination) which is simply the product of the number states of each component involved.

For the computers, the Markov finite state diagram is similar to the finite state machine of the MC and BC. Thus, Markov transition diagram for each computer consists of the following five states [7]:

1. *Computer operating*: the computer is operating correctly
2. *Loss of one input*: loss of input from one sensor of a monitored variable (e.g., the computer detects 0.0 vdc in the sensor reading or a value which is out of range for the measured variable or a physically impossible rate of change)
3. *Loss of both inputs*: loss of input from both sensors
4. *Arbitrary output*: the computer is sending random output to the controllers

5. *Computer down*: the computer recognizes a communication failure between itself and the sensors and the computer and takes itself down. Consequently, the computer is not sending any output to the controllers

The Markov models for the MC and BC are represented in Fig.6.1 and Fig.6.2 respectively.

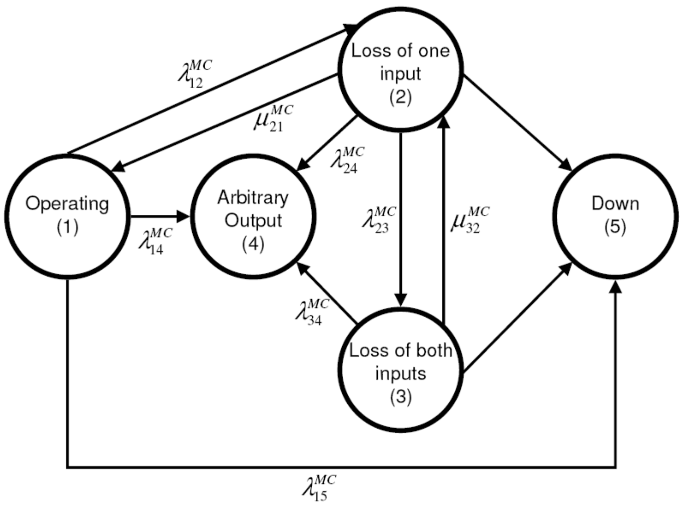


Fig.6.1 – Markov transition diagram for the MC

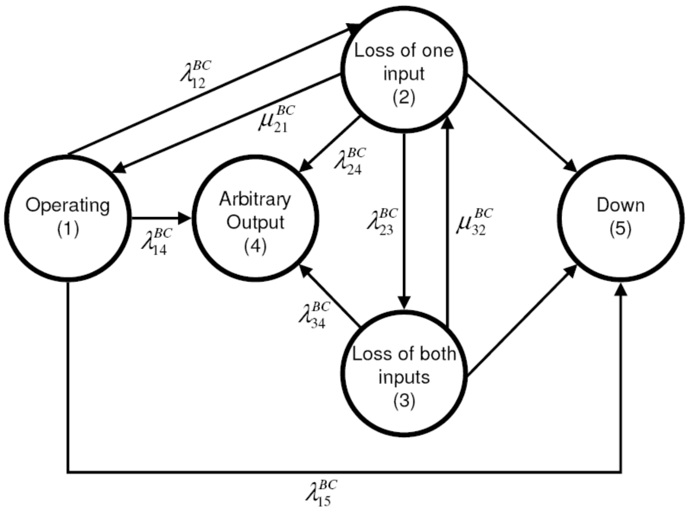


Fig.6.2 – Markov transition diagram for the BC

6.2 MFV, BFV and FP Controller

Section 4.4 shows how it is possible to merge two or more components into macro-components in order to reduce the computational efforts of the Markov/CCMT analysis. In this respect, it is convenient to merge the controllers and their associated activated devices (e.g., the MFV controller and the MFV). From Section 4.4 it is possible to identify the following eight states for the combined controller-actuated device Markov models [7]:

1. *Controller* is operating and *communicating with the actuated device*
2. *Controller* is operating and *not communicating with the actuated device*
3. *Freeze*: the controller is sending the last valid value to the actuated device
4. *Output high*: the controller is sending the highest value to the actuated device

5. *Output low*: the controller is sending the lowest value to the actuated device
6. *Arbitrary output*: the controller is sending a random value to the actuated device
7. *0 vdc output*: permanent no communication between controller and actuated device. 0 volts are detected from the actuated device
8. *Stuck*: actuated device is stuck in the previous position

Figure 6.3 shows the Markov model for the ensemble BFV-BFV controller which is conceptually identical to the combined MFV-MFV controller and FP-FP controller.

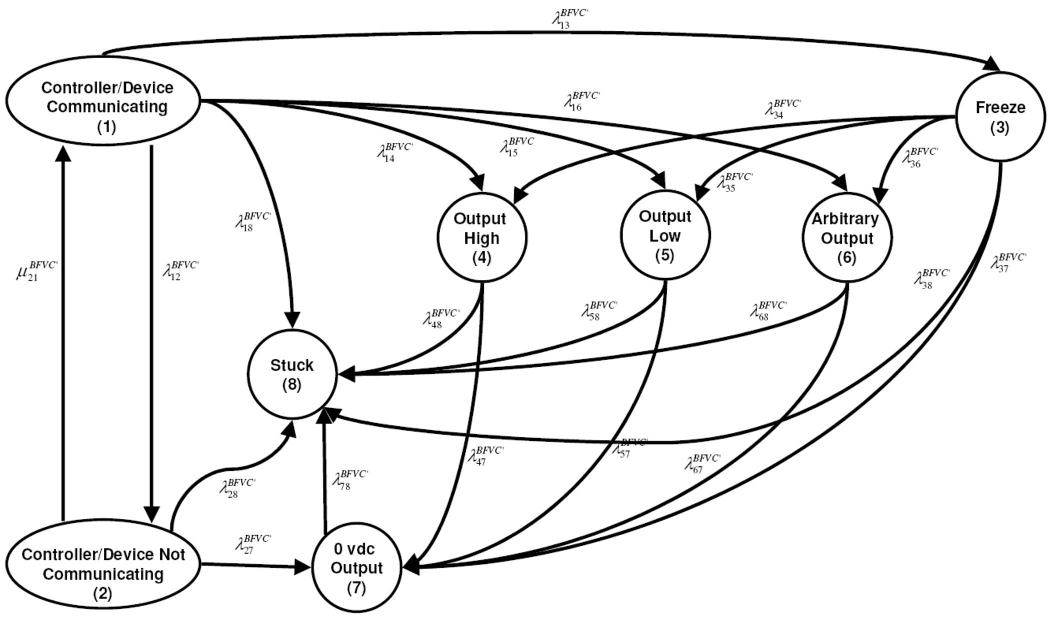


Fig.6.3 – Markov transition diagram for the combined BFV-BFV Controller

6.3 PDI Controller

The PDI controller has been modeled taking into account the failures presented in Section 2.5 and in Table 2.5.1 which are listed below [7]:

- Loss of inputs
- Loss of power
- Loss of outputs
- Arbitrary failure

Then it is possible to define the following states:

- *Operating*: PDI controller operating correctly.
- *Loss of inputs*: PDI controller does not receive any signal from the MFV controller
- *Arbitrary failure:* PDI controller is sending random value to the MFV

From Section 2.1, the loss of communication between the PDI controller and the MFV can be recovered. This recoverable failure (which introduces the need to remember the state of the components before the failure) can be modeled in a similar way as was done for the controllers (see Section 6.2); that is by duplicating the 3 states presented earlier (operating, loss of inputs and arbitrary failure). This approach results in:

- 3 states (operating, loss of inputs and arbitrary failure) when MFV and PDI are communicating correctly and,
- another set of 3 states (operating, loss of inputs and arbitrary failure) when MFV and PDI are not communicating.

The resulting model for the PDI controller is shown in Fig.6.4.

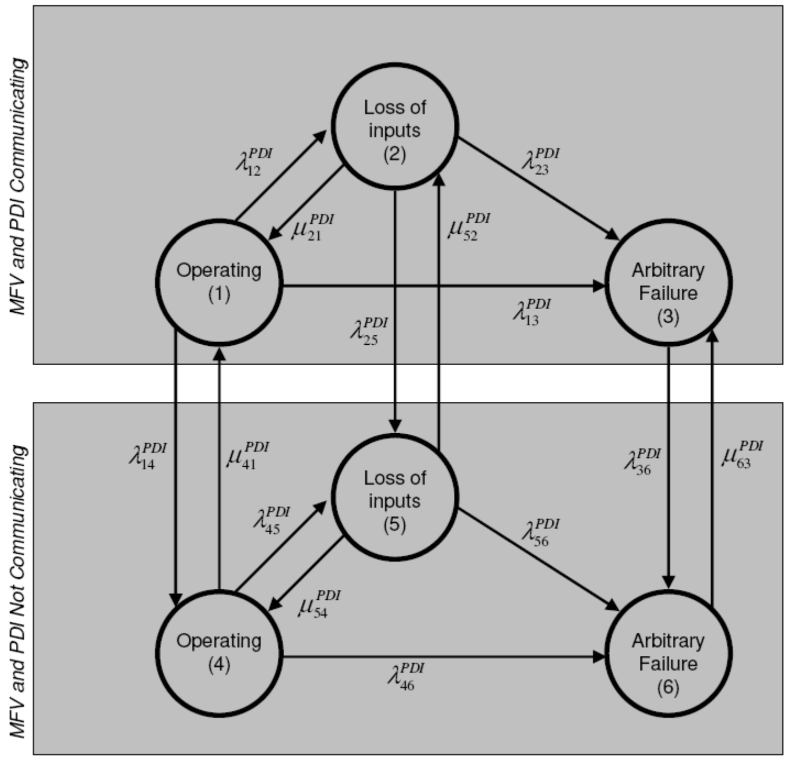


Fig.6.4 – Markov transition diagram for the PDI Controller

6.4 System state recombination

During the construction of the Markov transition diagrams for the components of the system under consideration, it is important to make sure that transitions in the Markov diagrams of a component are not dependent on another transition or a state of a different model. Such an independence is very useful to determine the component state transition probabilities as presented in Section 7.6.

In this respect, for the DFWCS system description presented in Section 2.1, there are 3 main issues which must be taken into account:

- The interactions between MC and BC
- The interactions between MC/BC and the MFV/FP/BFV controllers
- Commonality of power sources, one in common for MC and BC, one in common for MFV/FP/BFV controllers.

Sections 6.4.1 though 6.4.3 address these issues, respectively

6.4.1 Interactions between MC and BC

As shown in Section 2.1, the MC and BC transitions are coupled since one can take over if the other is failed. Since they are identical, the two computers have been merged into a single macro-computer. As a result of this combination, the MC and BC can be represented in a single Markov transition diagram with 15 states instead of 25 (5 state for the two computers: 5·5 = 25 states) as presented in Fig.6.5.

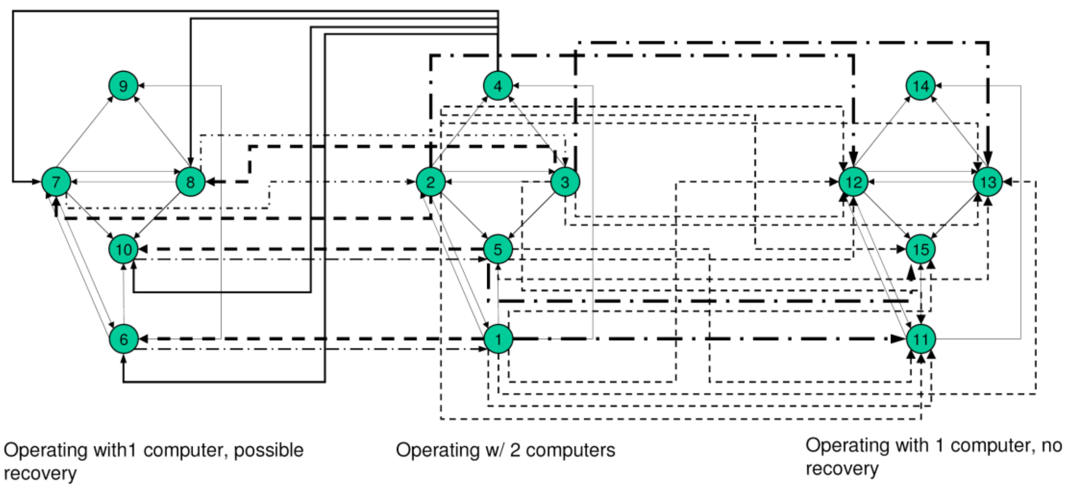

9
7
8
10
6
4
2
3
5
1
14
12
13
15
11
Operating with1 computer, possible recovery
Operating w/ 2 computers
Operating with 1 computer, no recovery

Fig.6.5 – Markov transition diagram for the combined system MC-BC

6.4.2 Interactions between MC/BC and the MFV/FP/BFV controllers

As presented in Section 4.4, when both the MC and BC are down, the MFV, BFV and FP controllers freeze their output. However, looking at Fig.6.3, there is not a single "Freeze" state in common for all the 3 controllers but there is "Freeze" state one for each controller. The goal is to obtain Markov transition diagrams for the computers and the controllers such that the transition to the "Freeze" is in common for all the controllers. Thus, the problem consists into the decoupling of the Markov transition diagrams of the 3 controllers.

This can be accomplished by removing the "Freeze" states from the MFV, BFV, and FP controllers transition diagrams and inserting a single "Freeze" state into the computer diagram as shown in Section 6.4.1 (see Fig.6.5). The Markov transition diagrams for the computer and for the combined BFV-BFV controller are shown in Fig.6.6 and Fig.6.7 respectively. The combined MFV-MFV controller and FP-FP controller the Markov models are similar to that of the combined BFV-BFV controller.

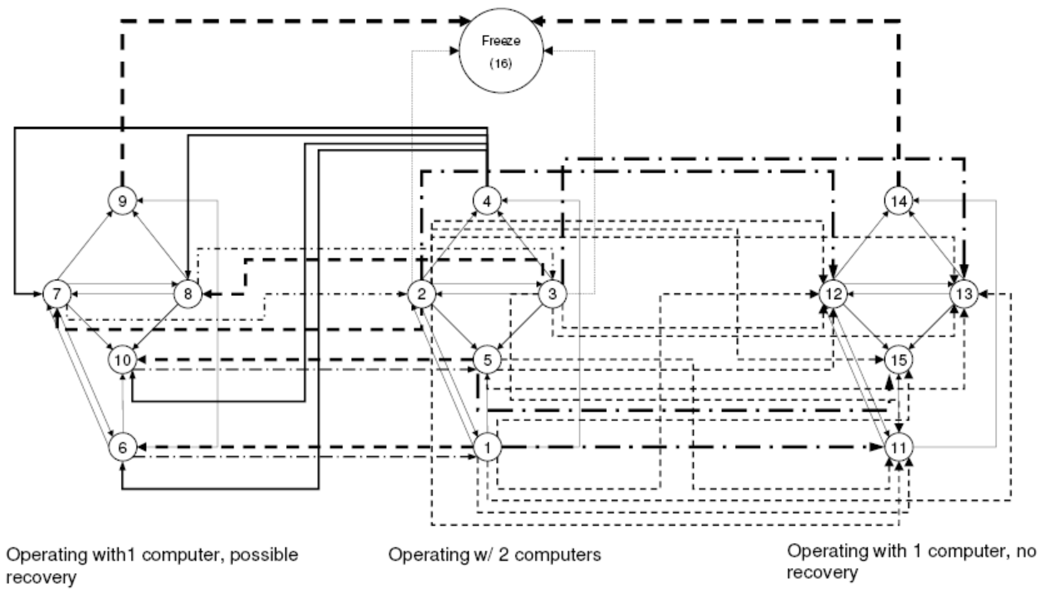

Freeze
(16)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Operating with1 computer, possible recovery
Operating w/ 2 computers
Operating with 1 computer, no recovery

Fig.6.6 – Markov transition diagram of the combined system MC-BC after the recombination

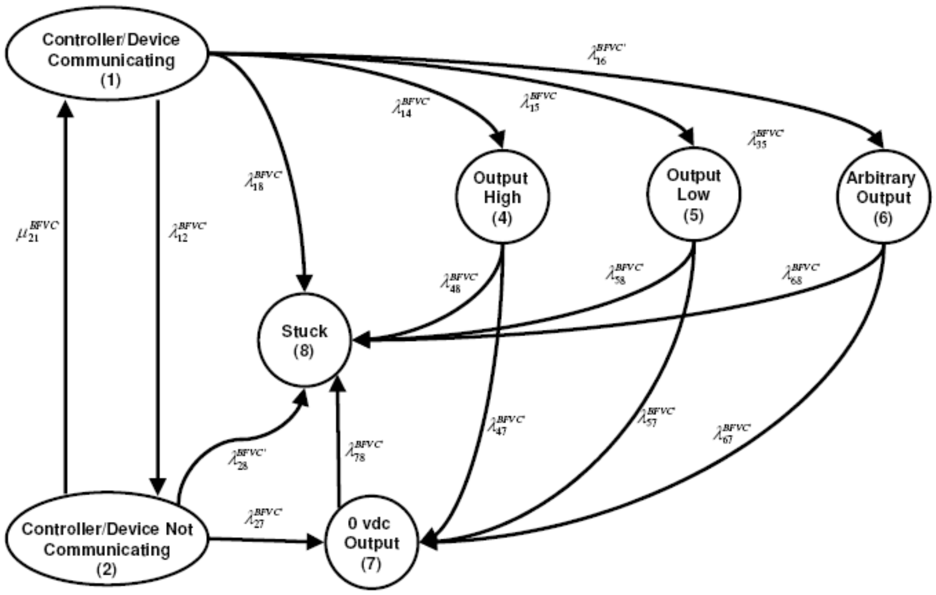


Fig.6.7 – Markov transition diagram of ensemble BFV-BFV controller after the recombination

6.4.3 Power sources

The two computers (MC and BC) and the three controllers (MFV, BFV, and FP) share the different power sources (Chapter 2). Thus, a failure in the power source of the computer or the controller, affects all the computers or all the controllers respectively.

In the Markov transition diagram of the controllers, a failure in the power source is represented through a transition to the “Output Low” state and to the “0 vdc output” state (sees Fig.6.7). This common cause failure can be modeled as follows:

1. introducing a separate Markov model for the power source of the controllers and,
2. removing the transitions to the “Output low” and “0 vdc output” states (see Fig.6.7) due to loss of power failure only

Then transition to the “Output low” state for the controllers is now due only to the “Low failure” of the controller according to , , .

The controller power source can be modeled simply as a two state Markov transition diagram where the states are the following:

- *Operating:* controllers power source active and,
- *No power*: no power is provided to the controllers.

The resulting model is shown in Fig.6.8.

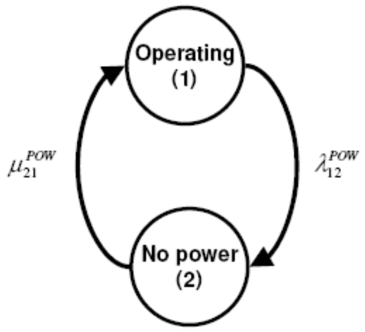


Fig.6.8 – Markov transition diagram for the controller power source

In 1 and , a failure in the computer power source causes the failure of both computers. Thus, the controllers freeze their outputs. This implies that a failure of the computers power source can be represented with a transition to the “Freeze” state (see Fig.6.1 and Fig.6.2). In this case a separate model is not necessary since the Markov model for the computers include both the MC and the BC. However, from every state of Fig.6.6, a transition to the “Freeze” state has to be added.

6.5 Component state combination reduction

Based on the discussion above, the number of the states of the Markov transition diagrams for each of the components of the DFWCS is as shown in .

Components	*# states*
Computers (Fig.6.6)	15
MFV Controller (Fig.6.7)	7
BFV Controller (Fig.6.7)	7
FP Controller (Fig.6.7)	7
PDI Controller (Fig.6.4)	6
Controller power (Fig.6.8)	2

Table 6.1 – List of states for each component of the DFWCS after recombination

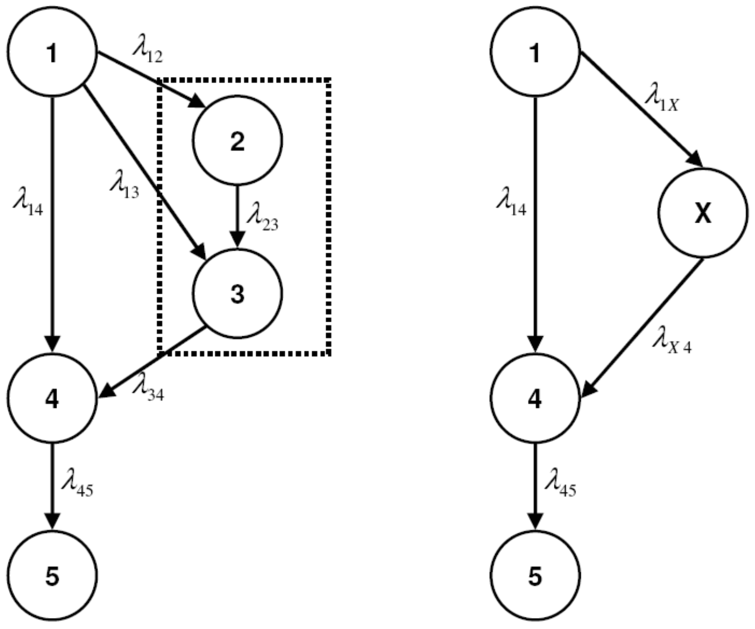
The resulting total number of component state combinations *N* is equal to the product of the states of all the components involved: $N = 15{\cdot}7{\cdot}7{\cdot}7{\cdot}6{\cdot}2 = 61740$.

Since the number of event sequences generated at every Markov time step increases exponentially with a base factor of *N*, it is necessary to reduce the number of components state combinations in order to obtain an analysis of the system with reasonable computing time. This reduction may be achieved by

1. reducing the number of components by merging two or more components together, and,
2. reducing the number of states of a component by merging two or more states together.

The first step has been already investigated and used in Section 6.2 and 6.4.1 for the controllers and for the computers respectively. The second step requires the merging of states. It is here chosen to merge all states in each Markov transition diagrams that have the same impact on the dynamics of the system.

As an example, the system of Fig.6.9 is considered. In this case, states 2 and 3 have the same impact and, thus, these states are merged.



(a) Original Markov transition diagram (b) Reduced Markov transition diagram

Fig.6.9– Illustrative example for the state reduction technique

Figure 6.9 shows how states 2 and 3 have been merged in a single state named X. After the merging process, transitions 1-X, and X-4 have to be determined. This is done as following:

1. *Solve the original Markov model: determine the state probabilities functions.* For the system of Fig.6.9, *$P_1(t)$, $P_2(t)$, $P_3(t)$, $P_4(t)$, $P_5(t)$* are determined first. Usually for small systems (e.g., Markov models with fewer than 5-6 states) it is possible to solve a system of first order differential equations symbolically using codes like Mathematica or solving it numerically with codes like Matlab. For systems with a greater number of states (i.e., more than 10) it is possible to solve a system of first order differential equations using matrices [2].
2. *Determine the new transition probabilities λ(t) of the reduced diagrams.*

The set of equations which describes the original Markov transitions diagram (Fig.6.9.a) are:

$$\left(\frac{dP_1(t)}{dt}\right)_a = -(\lambda_{12}+\lambda_{13}+\lambda_{14})\cdot P_1(t)$$

$$\left(\frac{dP_2(t)}{dt}\right)_a = -(\lambda_{23})\cdot P_2(t)+(\lambda_{12})\cdot P_1(t)$$

$$\left(\frac{dP_3(t)}{dt}\right)_a = -(\lambda_{34})\cdot P_3(t)+(\lambda_{23})\cdot P_2(t)+(\lambda_{13})\cdot P_1(t)$$

$$\left(\frac{dP_4(t)}{dt}\right)_a = -(\lambda_{45})\cdot P_4(t)+(\lambda_{14})\cdot P_1(t)+(\lambda_{34})\cdot P_3(t)$$

$$\left(\frac{dP_4(t)}{dt}\right)_a = (\lambda_{45})\cdot P_4(t)$$

The set of equations which describes the reduced Markov transitions diagram (Fig.6.9.b) are:

$$\left(\frac{dP_1(t)}{dt}\right)_b = -\left(\lambda_{1X} + \lambda_{14}\right)\cdot P_1(t)$$
$$\left(\frac{dP_X(t)}{dt}\right)_b = -\left(\lambda_{X4}\right)\cdot P_X(t) + \left(\lambda_{1X}\right)\cdot P_1(t)$$
$$\left(\frac{dP_4(t)}{dt}\right)_b = -\left(\lambda_{45}\right)\cdot P_4(t) + \left(\lambda_{14}\right)\cdot P_1(t) + \left(\lambda_{X4}\right)\cdot P_X(t)$$
$$\left(\frac{dP_4(t)}{dt}\right)_b = \left(\lambda_{45}\right)\cdot P_4(t)$$

Assuming that all the transition rates of the original system of Fig.6.9 are constant with time, $\lambda_{x4}(t)$ can be found in the following manner:

1. *Determine the equations which describe the time evolution of state 4 for both the model a) and b) of Fig.6.9.* These equations are:

$$\left(\frac{dP_4(t)}{dt}\right)_a = -\left(\lambda_{45}\right)\cdot P_4(t) + \left(\lambda_{14}\right)\cdot P_1(t) + \left(\lambda_{34}\right)\cdot P_3(t) \qquad (6.2)$$

(Model (a) of Fig.4.11)

$$\left(\frac{dP_4(t)}{dt}\right)_b = -\left(\lambda_{45}\right)\cdot P_4(t) + \left(\lambda_{14}\right)\cdot P_1(t) + \left(\lambda_{X4}\right)\cdot P_X(t) \qquad (6.3)$$

(Model (b) of Fig.4.11)

2. *Compare the two expression and then determine the equation of $\lambda_{x4}(t)$, as shown in Eq.4.6.*

$$\left(\frac{dP_4(t)}{dt}\right)_a = \left(\frac{dP_4(t)}{dt}\right)_b \rightarrow (\lambda_{34})\cdot P_3(t) = (\lambda_{X4})\cdot P_X(t) \rightarrow$$
$$\rightarrow \lambda_{X4} = \lambda_{X4}(t) = \frac{(\lambda_{34})\cdot P_3(t)}{P_X(t)} = \frac{(\lambda_{34})\cdot P_3(t)}{P_2(t)+P_3(t)} \tag{6.4}$$

Equation 4.6 shows that $\lambda_{x4}(t)$ is essentially an averaging of the transition rates for the transitions 2-3 and 2-4 of Fig.6.9., i.e.

$$\left(\frac{d(P_1(t)+P2(t))}{dt}\right)_a = -(\lambda_{34})\cdot P_3(t) + (\lambda_{12})\cdot P_1(t) + (\lambda_{13})\cdot P_1(t)$$
$$\left(\frac{dP_X(t)}{dt}\right)_a = -(\lambda_{X4})\cdot P_X(t) + (\lambda_{1X})\cdot P_1(t) \tag{6.5}$$

Repeating the procedure presented before for the other transition rate of Fig.6.9 it is possible to get:

$$\lambda_{1X}(t) = \lambda_{12} + \lambda_{13} \tag{6.6}$$

$$\lambda_{X4}(t) = \frac{(\lambda_{34})\cdot P_3(t)}{P_2(t)+P_3(t)} \tag{6.7}$$

Note the new transition probabilities for the reduced model are time dependent.

6.5.1 Reduced Markov transition diagram for the computers

As mentioned in Section 6.5, it was chosen to merge the states of a Markov transition diagram which have the same impact of the controller behavior. For the computers (see Fig.6.6), the possible outputs are:

- *Correct values*: values of MFV, BFV aperture and FP speed calculated from the input received form the sensors (i.e., states 1, 6, and 11 of Fig.6.5),
- *Old value*: last correct values of MFV, BFV aperture and FP speed (i.e., states 2, 3, 4, 7, 8, 9, 12, 13, 14, and 16 of Fig.6.5),
- *Arbitrary value*: random generated value (i.e., states 5, 10, and 15 of Fig.6.5).

The resulting reduced Markov model for the computers is shown in Fig.6.10.

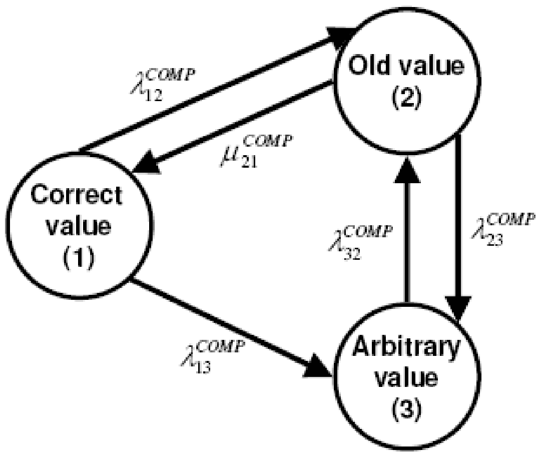


Fig.6.10 – Reduced Markov transition diagram for the computers

6.5.2 Combined Markov transition diagrams for the MFV, BFV and FP controllers

Looking at Fig.6.7, the possible outputs of the controllers are:

- *Correct value*: the controller is sending the correct values of MFV, BFV aperture and FP speed received by the computers (i.e., state 1 of Fig.6.7)
- *Old value*: the controller is sending the oldest correct values of MFV, BFV aperture and FP speed (i.e., state 8 of Fig.6.7)
- *Output high*: the controller is sending the highest values of MFV, BFV aperture and FP speed (i.e., state 4 of Fig.6.7)
- *Output low*: the controller is sending the lowest values of MFV, BFV aperture and FP speed (i.e., state 2, 5 and 7 of Fig.6.7)
- *Arbitrary output*: the controller is sending a random generated value of MFV, BFV aperture and FP speed (i.e., state 6 of Fig.6.7).

The resulting combined Markov model for the controllers is shown in Fig.6.11.

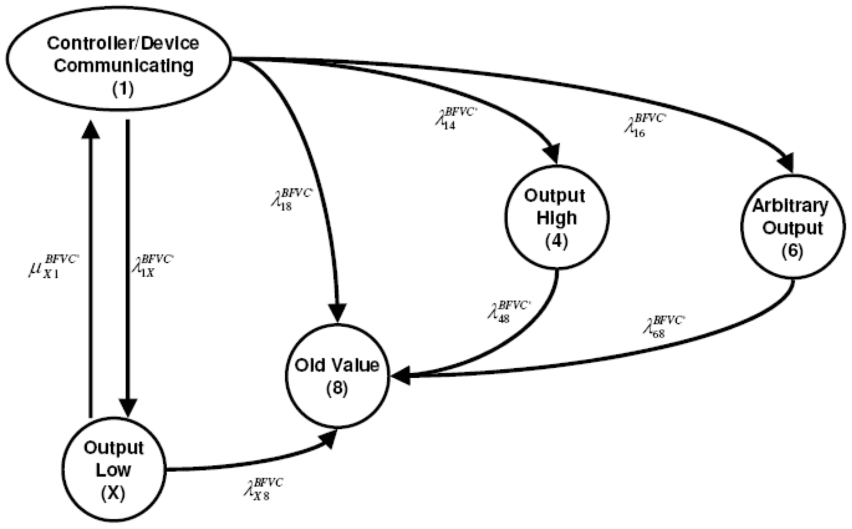


Fig.6.11 – Reduced Markov model for the controllers

6.5.3 Combined Markov transition diagram for the PDI controller

The possible outputs of the PDI controller are:

- Values of MFV aperture calculated from the computers (i.e., state 1of Fig.6.4),
- Last correct values of MFV aperture(i.e., state 2 Fig.6.4),
- Random generated value (i.e., state 3 of Fig.6.4),
- 0 vdc (i.e., state 4, 5 and 6 of Fig.6.4).

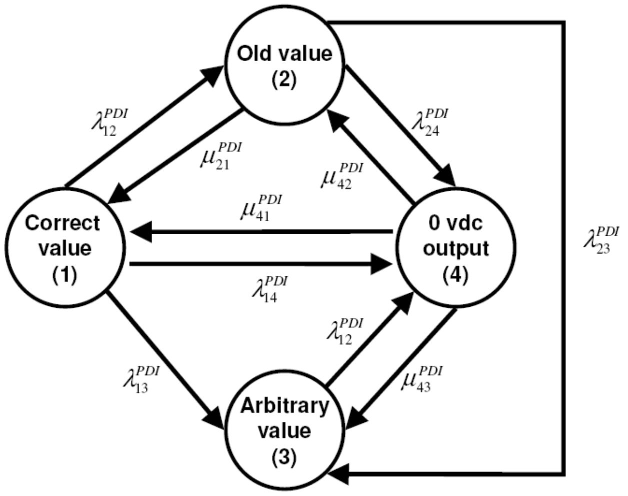


Fig.6.12 – Reduced Markov model for the PDI controller

6.6 Summary of the reduced Markov transition diagrams for the EIE

After the recombination (see Section 6.4) and reducing (see Section 6.5) processes, the overall number of components is considerably decreased from *N = 61740* (see Section 6.5 and) to *N = 3·5·5·5·4·2 = 3000* as shown in

Components	*# states*
Computers (Fig.6.10)	3
MFV Controller (Fig.6.11)	5
BFV Controller (Fig.6.11)	5
FP Controller (Fig.6.11)	5
PDI Controller (Fig.6.12)	4
Controller power (Fig.6.8)	2

Table 6.2 – List of component states after the recombination (see Section 6.4) and reduction processes (see Section 6.5)

CHAPTER 7

THE MARKOV/CELL-TO-CELL MAPPING TECHNIQUE

In the reliability model construction of digital I&C systems using the Markov/CCMT, the system failure probability (i.e., the probability that Top Events are reached) is evaluated throughout a series of discrete transitions within the controlled variable state space (CVSS). These discrete transitions take into account:

1. the natural dynamic behavior of the system (e.g. mass and energy conservation laws),
2. the control laws, and,
3. hardware/firmware/software states and their impact on the controlled/monitored process variables.

Items 1 and 2 above are modeled using the CCMT. Item 3 is modeled using a Markov transition diagrams presented in Section 6 for each of the components listed in Section 2. This section gives an overview of the mathematical foundation of the Markov/CCMT.

7.1 Construction of the Markov model using CCMT

The CCMT [16] is a systematic procedure used to describe the dynamics of both linear and non-linear systems in discrete time and in a discretized system state space (or the subspace of the controlled variables only). The CCMT first requires the partitioning of the state space or the CVSS into V_j ($j = 1,...,J$) cells (Section 7.3). Top events are represented as sink cells.

The evolution of the system is modeled and described through the probability $p_{n,j}(k)$ that the controlled variables are in a predefined region or cell V_j in the state space at time $t = k \cdot \Delta t$ ($k = 0,1,...$) with the system components (such as pumps, valves, or controllers) having a component states combination $n = 1,..,N$ (see Section 6.6). The state combination represents the system configuration at a given time and contains information regarding the operational (or the failure) status of each component.

Transitions between cells depend on:

- the dynamic behavior of the system
- control logic of the control system
- hardware/firmware/software states.

The dynamic behavior of the system is usually described by a set of differential or algebraic equations as well as the set of control laws, as presented in Section 2. The operating/failure states of each component are specified by the user. The procedure to determine:

- the cumulative distribution function (Cdf) of each Top Event,
- the probability distribution function (pdf) of each Top Event,
- the event sequences

follows several steps. These sequences of steps are explained in the following sections Section 7.2 to 7.8.

7.2 Definition of the Top Events

The controller is regarded as failed if water level in SGn rises above +30 inches and falls below -24 inches (see Section 2). Subsequently, there are two Top Events:

- $x_n < -24$ in (Low Level)
- $x_n > +30$ in (High Level).

The cells that correspond to Top Events are modeled as absorbing cells or sink cells, i.e., the system can not move out of these cells and thus the transition probabilities from these cells to others cells in the state space or CVSS are equal to 0.

7.3 Partitioning of the State Space or the CVSS into Computational Cells

The dynamics of the system is modeled as transitions between cells V_j ($j = 1,...,J$) that partition the state space or CVSS [7]. The partitioning needs to be performed in such a way that, other than V_j being disjoint and covering the whole space (definition of partitioning), values of the controlled variables defining the Top Events (in our case x_n) and the setpoints must fall on the boundary of V_j and not within V_j. Figure 7.1 graphically shows a possible partition of the CVSS of a system characterized by only three variables.

An important issue is the direct coupling between the discretization of the CVSS and the time step (Δt) of the simulation. In fact, given a discretization of the CVSS, a too low value of Δt could lead, in principle, to the system being unable to exit the starting cells unless properly chosen.

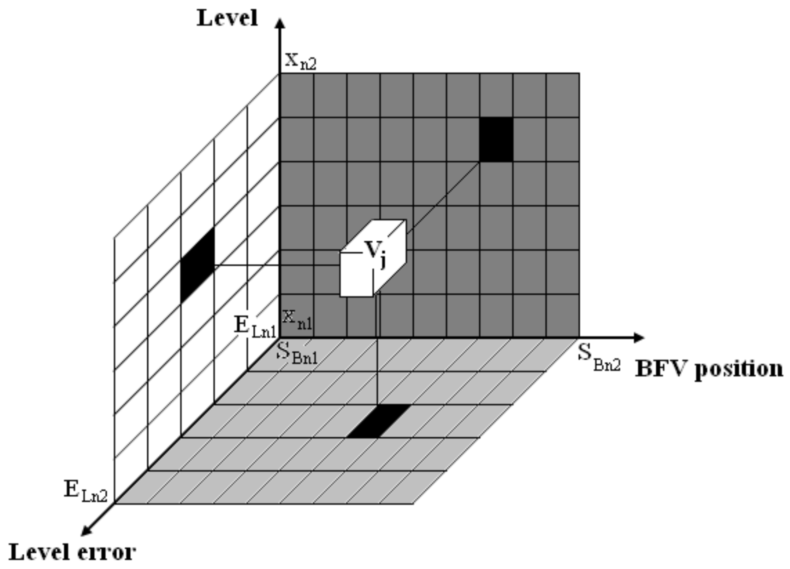


Fig.7.1 – Example of partitioning of the CVSS

7.4 Definition of the Hardware/Firmware/Software States

The definition of states in the construction of the Markov transitions diagrams for the components of the DFWCS described in Section 2 is presented in Section 6. In particular, Section 6.6 and summarize the final results for the Markov modeling of the DFWCS components.

7.5 Determination of Cell-to-Cell Transition Probabilities

As presented in Section 7.1, the evolution of any dynamic system depends on three factors essentially:

- the dynamic equations of the system,
- the control laws of the control system, and
- the state of each component.

Consequently, the probability of the system to transit from a cell $V_{j'}$ to cell V_j also depends on these three factors presented earlier. In the Markov/CCMT [7] [17], the first two factors are accounted for in the transition probability $g(j|j',n',k)$ while the third one is captured by the transition probability $h(n|n',j'\rightarrow j)$ (see Section 7.6).

The cell-to-cell transition probabilities $g(j|j',n',k)$ are conditional probabilities that the controlled variables are in the cell V_j at time $t = (k+1)\Delta t$ given that:

- the controlled variables are in the cell $V_{j'}$ at time $t = k\Delta t$, and,
- the system components are in component state combination $n(k) = n'$ at time t.

It can be shown that the $g(j|j',n',k)$ can be found from [7]:

$$g(j \mid j', n', k+1) = \frac{1}{\nu_j} \int_{V_j} d\nu' \, e_j\{\tilde{x}_{k+1}(x', n', k)\} \tag{7.1}$$

$$e_j\{\tilde{x}_k\} = \begin{cases} 1 \rightarrow \tilde{x}_k \in V_j \\ 0 \rightarrow otherwise \end{cases} \tag{7.2}$$

where:

- v_j is the volume of the cell V_j
- $\tilde{x}_k$ is the arrival point in the state space/CVSS at time $t = (k+1)\Delta t$
- x' is the starting point in the cell $V_{j'}$ at time $t = k\Delta t$
- n' is the component state combination at time $t = k\Delta t$.

The algorithm to determine $g(j|j',n',k)$ is the following:

1. Partition a cell j' into N_p subcells.
2. Choose the midpoint of each subcell, integrate the equations which describe the dynamic behavior of the system (e.g., for the DFWCS the equations are presented in Section 2.2.2) over the time interval $k\Delta t \leq t \leq (k+1)\Delta t$ under the assumption

that the component state combination remains n’ at all times during $k\Delta t \leq t \leq (k+1)\Delta t$.

3. Observe the number of arrivals in N_p+1 at time $t = k\Delta t$ (i.e., $\tilde{x}_{k+1}(x', n', k)$).
4. Obtain $g(j|j',n',k) = N_p / (N_p+1)$.

This process is equivalent to approximating the integral in Eq.(7.1) by an N_p point, equal weight quadrature scheme.

7.6 Determination of Hardware/Firmware/Software State Transition Probabilities

The stochastic behavior of hardware/software/firmware is represented through $h(n|n',j'\rightarrow j)$, which is the probability that the component state combination (see Section 6.6 and) at time $t = (k+1)\Delta t$ is n, given that:

- $n(k) = n'$ at $t = k\Delta t$, and
- the controlled variables transit from cell $V_{j'}$ to cell V_j during $k\Delta t \leq t < (k+1)\Delta t$.

For components with statistically independent failures, the probabilities $h(n|n',j'\rightarrow j)$ are the products of the individual component failure or non-failure probabilities during the mapping time step from $k\Delta t$ to $(k+1)\Delta t$, i.e.,

$$h(n\,|\,n', j' \rightarrow j) = \prod_{m=1}^{M} c_m(n_m\,|\,n'_m, j' \rightarrow j) \tag{7.3}$$

where $c_m(n_m|n'_m,j'\rightarrow j)$ is the transition probability for component m from the combination n'_m to n_m within $[k\Delta t, (k+1)\Delta t]$ during the transition from the cell $V_{j'}$ to V_j. Each $c_m(n_m|n'_m,j'\rightarrow j)$ is thus determined from the transitions diagrams presented in Section 6.

It is important to highlight that after the recombination and reducing processes presented in Sections 6.4 and 6.5, respectively, the transition probabilities of the DFWCS components are time dependent.

7.7 Construction of the Cell-to –Cell Transitions Probabilities

The definition of the two transition probabilities $h(n|n',j'\rightarrow j)$ and $g(j|j',n',k)$ consent to determine the evolution of the system in terms of the probabilities $p_{n,j}(k+1)$. In particular, the probability $p_{n,j}(k+1)$ that, at $t = (k+1)\Delta t$, the controlled variables are in cell V_j and the component state combination is n is the sum of $N\times J$ terms where N is the total number of component state combinations and J tis the total number of cells which partition the CVSS.

Each of these terms is the product of two factors:

- the probability $q(n,j|n',j',k)$ for the system to transit from the cell $V_{j'}$ and component state combination n' to cell V_j and component state combination n, and,
- the probability that the system is in the initial cell $V_{j'}$ and state combination n' ($p_{n,j}(k)$).

Thus:

$$p_{n,j}(k+1) = \sum_{n'=1}^{N}\sum_{j'=1}^{J} q(n, j \mid n', j', k) p_{n',j'}(k) \tag{7.4}$$

The elements of the transition matrix *q(n,j|n',j',k)* are functions of both

- the cell to cell transition probability *g(j|j',n',k)* (see Section 7.5), and
- the component state transition probability *h(n|n',j'→j)* (see Section 7.6)

From [14]:

$$q(n, j \mid n', j', k) = g(j \mid n', j', k) h(n \mid n', j' \to j) \tag{7.5}$$

Since cells V_j cover the whole CVSS and *N* includes all the possible state combinations:

$$\begin{gathered} \sum_{n'=1}^{N}\sum_{j'=1}^{J} q(n, j \mid n', j', k) = 1 \\ \sum_{n'=1}^{N}\sum_{j'=1}^{J} p_{n',j'}(k) = 1 \end{gathered} \tag{7.6}$$

7.8 Determination of pdf and Cdf

After computing *$p_{n,j}(k)$* at the end of each time step *k*, the Cdf *$F_\gamma(k)$* and pdf *$w_{n,\gamma}(k)$* for the TopEvent γ (γ = 1,..., Γ) can be calculated as presented in Eq.7.8:

$$\begin{aligned} F_\gamma(k) &= \sum_{n=1}^{N} p_{n,\gamma}(k) \\ w_{n,\gamma}(k) &= \frac{1}{\Delta t}\left[p_{n,\gamma}(k+1) - p_{n,\gamma}(k)\right] \end{aligned} \tag{7.7}$$

Each Top Event γ is a set of cells in the CVSS.

Statistical importance of hardware/software/firmware configuration *n* to failure event γ at time t= kΔt can be determined from:

$$(\mathrm{Im})_{n,\gamma}(k) = \frac{\sum_{n\in S_n} w_{n,\gamma}(k)}{\sum_{n=1}^{N} w_{n,\gamma}(k)} \tag{7.8}$$

where S_n is the set of system states containing the hardware/software/firmware configuration *n*.

CHAPTER 8

EXAMPLE INITIATING EVENT ANALYSIS

This section shows how it is possible to implement the Markov/CCMT methodology presented in the previous sections. The system under consideration is presented in Section 8.1 and it is a simplified version of the DFWCS presented in Section 2. The implementation of the Markov/CCMT methodology has been accomplished with a code written in Java and presented in Section 8.2 and in Appendix A. Since, the code SAPHIRE requires as input are dynamic event trees (DET) [18], Markov/CCMT produce event sequences which can be easily interpreted as dynamic event trees (DET) as shown in Section 8.3.

8.1 An example initiating event (EIE)

Most of the analysis performed for Level 2 PRA assumes that the reactor is shutdown in all the initiating events. In this respect, the following initiating event (EIE) [7] is used in order to illustrate how it is possible to implement Markov/CCMT methodology for the reliability analysis of digital control systems:

1. Turbine trips
2. Reactor is shutdown
3. Power *P(t)* is generated from the decay heat
4. Reactor power and steam flow rate decay from 6.6% of 3000 MW$_{th}$ (or 1500 MW$_{th}$/SG) and the analysis starts10 second after reactor shutdown
5. Feedwater flow and level are initially at nominal value
6. Off-site power is available
7. Main computer is failed.

In summary, the reactor is originally at full power (3000 MW$_{th}$)., Following the turbine trip (initiating event), the reactor is shut down. Ten seconds after shutdown, the decay power for a reactor operating previously for one year at full power (3000MW$_{th}$) is [13]:

$$P(t) = 0.66 \cdot 3000 \cdot \left[\frac{1}{(10+t)^{0.2}} - \frac{1}{(3.15 \cdot 10^{7} + t)^{0.2}} \right] \tag{8.1}$$

From Section 2, the control laws needed for the EIE are the following:

Rate of level change: $$\frac{dx_n}{dt} = A \cdot (f_{wn} - f_{sn}) \tag{8.2}$$

Compensated water level: $$\tau_2 \frac{dC_{Ln}}{dt} = -C_{Ln} + x_n + \tau_1 \cdot (f_{wn} - f_{sn}) \tag{8.3}$$

Compensated power: $$\tau_4 \frac{dC_{pn}}{dt} = -C_{pn}(t) + p_n + \tau_3 \frac{dp_n}{dt} \tag{8.4}$$

Flow Demand (Low Power):

$$C_{Bn}(t) = \nu_{Bn}\alpha_M + \nu_{Bn}C_{Pn} + \beta_{Bn}(h_{wn})\int dt[r_n - C_{Ln}(t)] - \lambda_{Mn}(\sigma_{Mn}) \tag{8.5}$$

BFV Demand: $$\sigma_{Fn}(t) = \begin{cases} C_{Bn}(t) & \text{(Low Power)} \\ 0 & \text{(High Power)} \end{cases} \tag{8.6}$$

FP Demand: $$\sigma_{Fn}(t) = \begin{cases} \sigma_{Fn}(\mathrm{C_{Fn}} = 0) & \text{(Low Power)} \\ \sigma_{Fn}(\max(\mathrm{C_{Fn}}, \sigma_{Mn}^{-1}(\mathrm{C_{Fn}}))) & \text{(High Power)} \end{cases} \tag{8.7}$$

BFV Position: $$\tilde{S}_{Bn}(t) = \begin{cases} \sigma_{Bnm} & \text{MC operating} \\ \sigma_{Bnb} & \text{BC operating, MC down} \\ \eta_{Bn} & \text{BC down, MC down} \end{cases} \tag{8.8}$$

The feedwater flow is obtained from the solution of:

$$\frac{4.73\mathrm{L}(100/\tilde{S}_{Bi})^2 f_{wn}^{1.852}}{\mathrm{C}^{1.852}\mathrm{D}^{4.87}} = 136 + 6.3\times10^{-6} f_{wn} - 4.6\times10^{-11} f_{wn}^2 \tag{8.9}$$

For the example initiating event, the state space is 4-dimensional (see Fig.7.1) and is comprised of

- level x_n
- level error E_{Ln}
- compensated level C_{ln}
- BFV position S_{Bn}.

As presented above, the components involved in the EIE are BC, BFV and BFV controller. In order to simplify the overall system presented in Sections 2 and 4 for illustration purposes, the following assumptions are introduced:

1. Only loss of both inputs can occur (and not possibly one).
2. Only the BFV controller failure can generate arbitrary output. If BC generates arbitrary output due to internal failure, it is recognized by the BC and BC transits to State D.
3. Loss of communications between the sensors and BC and between BC and BFV controller cannot be recovered.
4. The BFV controller cannot fail in Output High mode.
5. FP cannot fail.

The BFV controller Output Low mode failure is included in 0 vdc Output in Fig.6.7 since they have the same effect on the actuated device (i.e. valve totally closed). Also, since the MC has failed (not recoverable) and only the BC computer is operating, the computer-computer connections (see Fig.6.6) are reduced to the one presented in MS 3 only. Due to Assumptions 1 and 2, Loss of One Input and Arbitrary Output (see Fig.6.5) are not considered in intra-computer interactions.

From Assumption 3, the Plane 2 that represents the loss of output of the controller in Fig.6.7 is no longer needed. Subsequently, Loss of Output leads the controller to transit to the 0 vdc Output directly.

Figure 8.1 shows the finite state machine model of the DFWCS for the EIE. The states of this Markov transition diagram are:

- *Controller/Device Communicating*: This macro state indicates that the combined BFV-BFV controller is operating correctly and it contains the following BC states:

    - State A: BC operating correctly
    - State B-C: BC does not receive any signals from the sensors (BC detects vdc in the sensor reading)
    - State D: BC down

- *Freeze*: BFV controller recognizes that BC is down and maintains the position of the BFV.
- *Stuck*: BFV remains stuck in the same position due to mechanical failure.
- *Arbitrary Output*: BFV controller is sending random data to the BFV due to internal failure (software/firmware/hardware).

- *0 vdc Output*: 0.0 vdc on the line which connects controller and valve. This can be due to both a loss of communication between BFV controller and BFV or also due to the Low Output mode failure of the BFV controller.

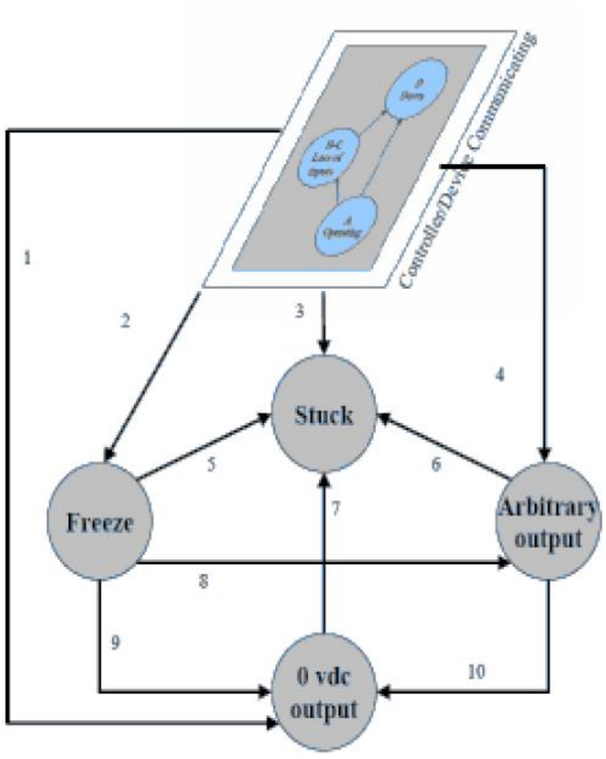


Fig.8.1 – Finite state machine for the EIE

A list of the possible transitions between these states is presented in Table 8.1

.

Failure mode	Transitions
BFV stuck due to mechanical failure	3, 5, 6, 7
BFV controller drifts low or 0 vdc on the line	1, 9, 10
BC down due to internal failure (e.g. watchdog timer), power loss or loss of output	2
Random output from the BFV controller to the BFV	4, 8

Table 8.1 – Possible transition for the EIE

The BFV position is function of both the BC and BFV controller states and may reflect history dependence. In this respect, shows the BFV position as function of the possible components state combinations.

n	BFV Controller	BC	BFV Position
1	Controller/Device Communicating	State A	Calculated from Eqs. (2.5.17)-(2.5.21)
2	Controller/Device Communicating	State B-C	Maintained
3	Controller/Device Communicating	State D	Maintained
4	Freeze	---	Maintained
5	Arbitrary Output	---	Random in the range 0-100%
6	0 vdc Output	---	Closed
7	Stuck	---	Maintained

Table 8.2 – BFV position as function of the system state for the EIE

8.2 Program description

The implementation of the Markov/CCMT applied to the EIE presented in Section 8.1 has been accomplished by building a program written in Java and presented in Appendix A. Java is an object-oriented language developed by Sun Microsystems in the mid ‘90s. This language derives great part of its syntax from the C language and it has great capabilities to deal with objects. This feature is particularly useful for the implementation of the Markov/CCMT methodology.

The main idea of the program is to build a “dynamic’ array of objects where each object is an event sequence. As mentioned in Section 7, each event sequence is the trajectory of a single point in the CVSS (see Fig.7.1). In computing language, an array is simply a vector of objects with a finite length fixed by user at the moment of its declaration. The concept of dynamic arrays has been introduced to compensate the need of an array whose length is a variable and where new objects can be inserted or deleted easily. Each object, in this case, is representing the trajectory of a point in the CVSS during the mission time.

The mission time is divided into “time steps” where each time step has the same time length. During this time step, also called Markov time, the component state combination remains untouched.

Starting from a fixed number of initial points in a cell of the CVSS (see Fig.7.1), the program starts to follow the trajectories of these points. As presented in Section 7, the transition from a cell to another depends on the component state combination and the dynamic of the system (i.e., the control laws).

In Sections 8.2.1 - 8.2.3 the program is described in detail.

8.2.1 Inputs

The inputs needed for the Markov/CCMT analysis include the information on both Type I and Type II interactions as shown in Fig.1.1 and in Section 1.

Type I interactions relate to indirect interactions between the system components through the controlled/monitored process and have been presented in Section 2. These interactions are modeled by the set equations presented in Section 8.1 and the numerical implementation shown in Section 8.2.3.

Type II interactions relate to direct interactions between the components of the control system. These interactions have been modeled using Markov transition diagrams as presented in Section 6. Section 8.1 shows the finite state machine for the EIE. The resulting Markov transition diagram has the same number of states as the finite state machine and transitions between these states can be described using a matrix *M*. The matrix *M* for the EIE is presented in . Each element *m*[*i*,*j*] of *M* indicate the probability of transition from state *i* to *j*. Since failure data are not available, *m*[*i*,*j*] is 1 if the transition is not possible and 0 otherwise.

State	*1*	*2*	*3*	*4*	*5*	*6*	*7*
1	1	1	1	1	1	1	1
2	0	0	1	1	1	1	1
3	0	0	0	1	1	1	1
4	0	0	0	1	1	1	1
5	0	0	0	0	1	1	1
6	0	0	0	0	0	1	1
7	0	0	0	0	0	0	1

Table 8.3 – Transition scheme for the EIE

The level range was partitioned into three cells, one cell corresponding to the normal operating range and the other two cells corresponding to the two Top Events, High Level and Low Level.

8.2.2 Classes and objects

In several programming languages it is possible to define more complicated structures (known also as classes) other than the elementary structures such as characters, numbers or Boolean. The purpose of these structures is to incorporate several variables of different formats (such as characters, numbers or Boolean) in a single entity: a class.

For the EIE (but, in general, for every Markov/CCMT analysis) the trajectory of each scenario is a sequence of several points where each point is characterized by the following information:

- the components state combination *n*
- the cell j in the CVSS

Figure 8.2 shows two event sequences (i.e., the trajectories of 2 points).

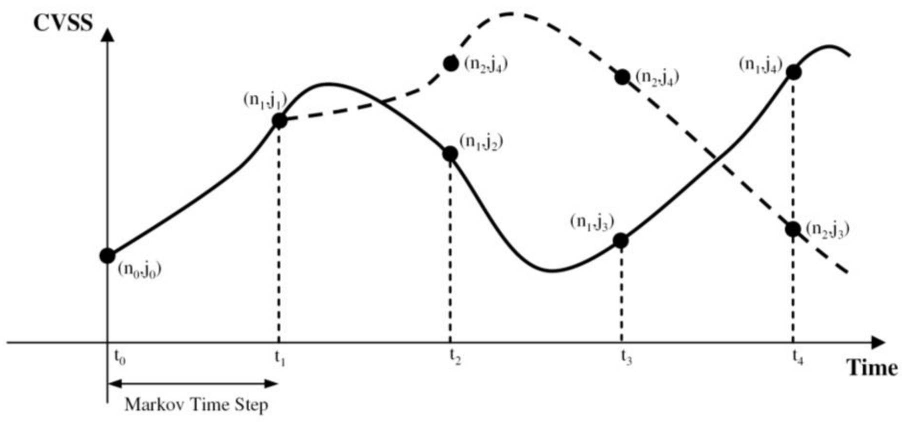


Fig.8.2 – Characterization of the dynamic of the points generated by the Markov/CCMT methodology

Moreover, as presented in Section 8.1, every cells *j* is characterized by the following variables:

- level
- compensated level
- level error
- BFV position

Note that since the whole normal operating range is characterized by one cell, all the arrival points $\tilde{x}_{k+1}$ and departure points x' in Eq.(7.1) that fall within the normal operating range are contained within the same cell. Subsequently, $g(j \mid j', n', k+1) = 1$ for *j'* corresponding to the normal operating range unless *j* corresponds to High Level or Low Level. In the latter two cases, $g(j \mid j', n', k+1) = 0$ for *j*' and *j* corresponding to the normal operating range.

Figure 8.2 shows how it has been possible to implement a point using Java thorough a specific class called "step" which incorporate the following variables:

- the component state combination *n*
- level
- compensated level
- level error
- BFV position

```
public class step
{
      public int n; // component state combination

      public double l; // level
      public double cl; // compensated level
      public double le; // level error
      public double bfvpos; // BFV pos

      public step ()
      {
            n = 0;
            l = 0;
            cl = 0;
            le = 0;
            bfvpos = 0;
      }

public step (int stateComb, double level, double compensatedLevel,
                  double levelError, double bfv)
      {
            n = stateComb;
            l = level;
            cl = compensatedLevel;
            le = levelError;
            bfvpos = bfv;
      }
}
```

Fig.8.3 – Class implemented in Java to define a point in CVSS for the EIE

A single event sequence is simply a sequence of points and is implemented in Java as an array of "step". This array is called "branch" and it is shown in Fig.8.4.

```
public class branch
{
      public step [] path ;

      public int completedSteps;

      public branch (int tSteps)
      {
            path = new step [tSteps];

            completedSteps = 1;

            int initialStateCombination = 0;
            double initialLevel = 0;
            double initialCompensatedLevel = 0;
            double initialLevelError = 0.0;
            double bfvpos = 0.0;

            for (int k=0 ; k<tSteps ; k++)
            {
                  path[k] = new step(initialStateCombination ,
initialLevel, initialCompensatedLevel, initialLevelError, bfvpos);
            }
      }
}
```

Fig.8.4 – Class implemented in Java to define scenario for the EIE

8.2.3 Program Engine

The main purpose of the engine program, shown in Fig.8.5, is to follow and analyze the path of every point. This engine performs the following operations for every Markov time step and for each point in CVSS:

1. generate new points, one for each transition in the Markov model (see)
2. determine the new position of the original step and the position of the points generated in 1 using the EIE simulator (See Section 8.2.4).

The function which accomplishes this task is the function “nodeAnalysis” and it is shown in Fig.8.5 and entirely in Appendix A

```
for (int i = 0 ; i < tree.capacity() ; i++)
{
      temp = tree.elementAt(i);

      for (int j = temp.completedSteps-1 ; j<timesteps-1 ; j++)
      {
            tree = nodesAnalysis (tree, i, j, matrix_n, dt);
      }
}
```

Fig.8.5 – Program engine

8.2.4 EIE simulator

The simulator for the EIE is implemented directly in Java as a separate function. This function receives as inputs the following parameters:

- the initial point (n, j)
- the Markov time step Δt

The program, given the cell of the initial point, determines the value of the theoretical aperture of the BFV. Then, depending on the status of the components state combination, it determines the real aperture of the BFV and the consequent feedwater flow. Thus, the arriving cell *j’* is determined. The point generated (situated in cell *j’* and having the same component state combination *n*) is then saved in the array.

8.2.5 Output

The output of the program is a list of all the possible trajectories (i.e., a list of all the possible event sequences) given the set of initial points. Since the CVSS (level) is represented by a single cell, the Cdf for High Level (P_H) or Low Level (P_L) occurrence at time $t=k\Delta t$ can be found from

$$P_H = \sum_{k'=1}^{k} \sum_{n} \sum_{n'} q(n, H \,|\, n', j', k) \qquad \text{and} \qquad P_L = \sum_{k'=1}^{k} \sum_{n} \sum_{n'} q(n, L \,|\, n', j', k) \quad (8.10)$$

where *j'* corresponds to the normal operating range.

Event sequences and DET are logically equivalent. In fact, assuming that the system starts in an operational state where all process variables are in the nominal range and all system components are operational, the algorithm starts branching through discrete time steps such that each level of branching in the tree represents all the possible states in which the system may be after a given time interval. Branching stops any time a branch reaches a "sink" state (i.e., a state from which the system cannot move out) or the probability associated with the branch is below a chosen threshold. It is also possible to stop after a certain amount of system time has elapsed or, equivalently, once the branching has reached a chosen depth in the tree.

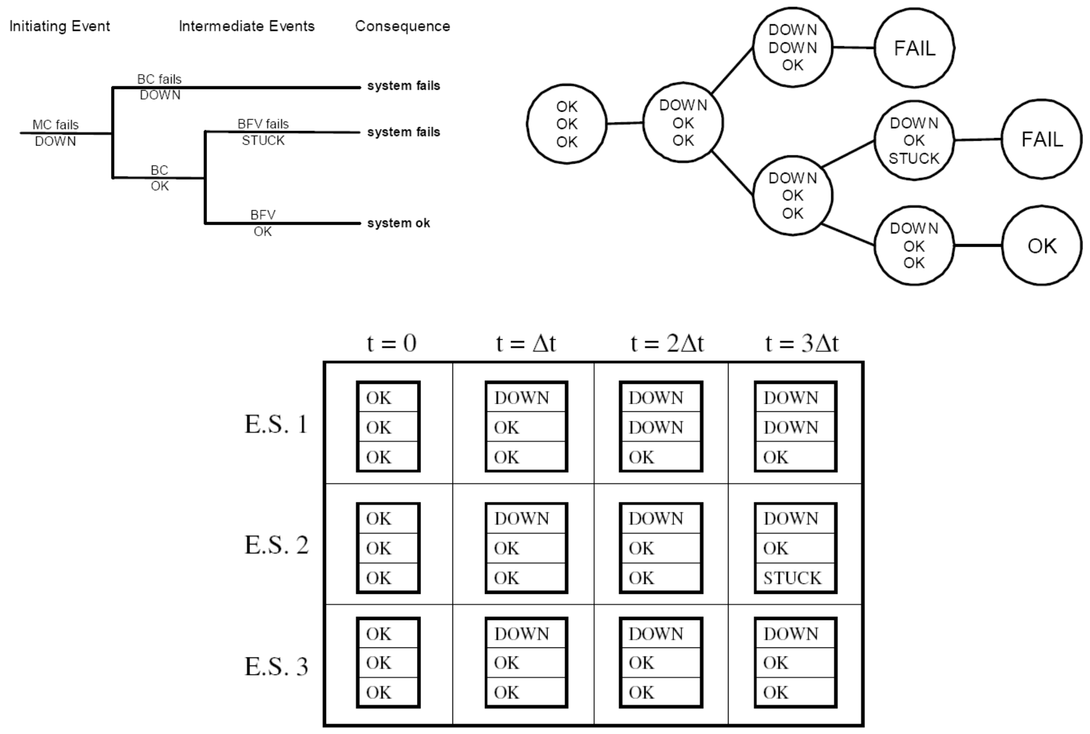


Fig.8.6 – Event trees data structure and event sequences

The DET is represented by a tree data structure. A tree data structure is composed of "nodes" (where information is stored) and "links" that connect the nodes. The nodes in the tree data structure correspond to the branching points in the DET and the links represent the branches. Figure 8.6 shows 3 event sequences (E.S. 1, E.S. 2 and E.S. 3), the relative event tree and a tree data structure that might be used to represent it. In the example, the tree nodes hold the information about the system component states: MC, BC and BFV. The event corresponding to a specific branch in the tree can be deduced by comparing the configurations at the beginning and at the end of the branch. Since the level is partitioned only into three cells, a large number of nodes will fall within the normal operating range for level. In this situation, it is also possible to regard the CVSS cells as the nodes. Then if the $g(j \mid j', n', k+1)$ obtained from the trajectories generated by the algorithms in Figs. 8.4 and 8.5 (see Section 8.2.2) are used to determine the links between these nodes, the links represent the expected behavior of the trajectories, weighted by the probability of occurrence of the trajectories as determined by $h(n \mid n', j' \rightarrow j)$.

8.3 Results

The tool used to generate event sequences starts from a normal state in which all the system components are operational and the process variables are within their nominal range.

Table 8.4 shows an example of failure scenario. For each time step are indicated the system configuration, the level of the SG and comments regarding the status of the DFWCS.

Time [s]	System configuration	Level [ft]	Comments
1	BC: Operating BFV: Communicating	0	Both BFV and BC are in their operational state, and all process variables are in their nominal range.
2	BC: Operating BFV: Communicating	-0.404	Both BFV and BC are in their operational state, and all process variables are in their nominal range.
3	BC: Loss of Input BFV: Communicating	0.534	BC loses and input from one sensor.
4	BC: Down BFV: Communicating	1.039	Since BC does not recover the loss of input, it takes itself down.
5	BC: Down BFV: Freeze	1.555	Since BC is down, controller BFV controller freezes its output.
6	BC: Down BFV: Arbitrary Output	2.610 (High)	Internal failure of the BFV controller which generate arbitrary output.

Table 8.4 – Example of scenario for the EIE

Table 8.5 summarizes the event sequences generated by the program for the EIE. In particular, it shows the number of scenarios generated for different time steps and the percentage of high and low scenarios. Moreover, Fig.8.7 and Fig.8.8 display the number of failure scenarios as function of time steps.

# time steps	*Tot*	*Low*	*High*	*% Low*	*% High*
0	0	0	0	0	0
1	1	0	0	0	0
2	7	0	0	0	0
3	26	0	0	0	0
4	70	0	0	0	0
5	155	0	1	0	0.6
6	301	0	12	0	4.0
7	532	45	22	8.5	4.1
8	876	80	37	9.1	4.2
9	1365	133	66	9.7	4.8
10	2035	259	137	12.7	6.7
11	2926	456	304	15.6	10.4
12	4082	739	510	18.1	12.5
13	5551	1064	803	19.2	14.5
14	7385	1629	1307	22.1	17.7
15	9640	2346	1885	24.3	19.6

Table 8.5 – Summary of the event sequences generated for the EIE

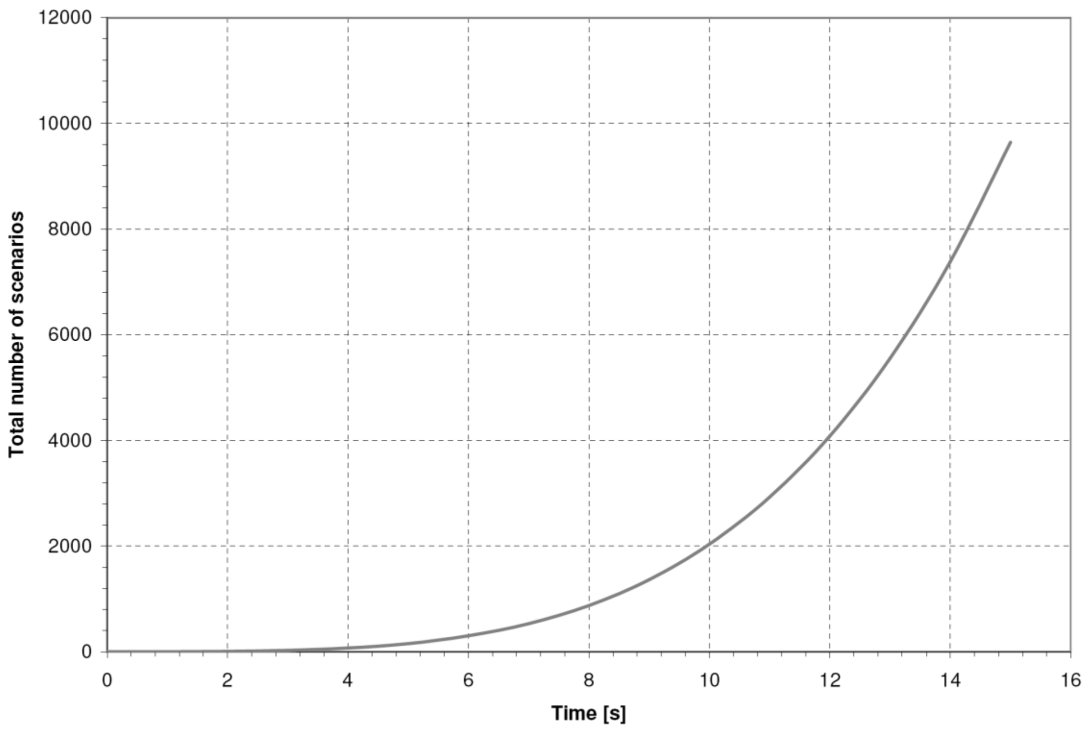


Fig.8.7 – Number of scenarios generated for the EIE

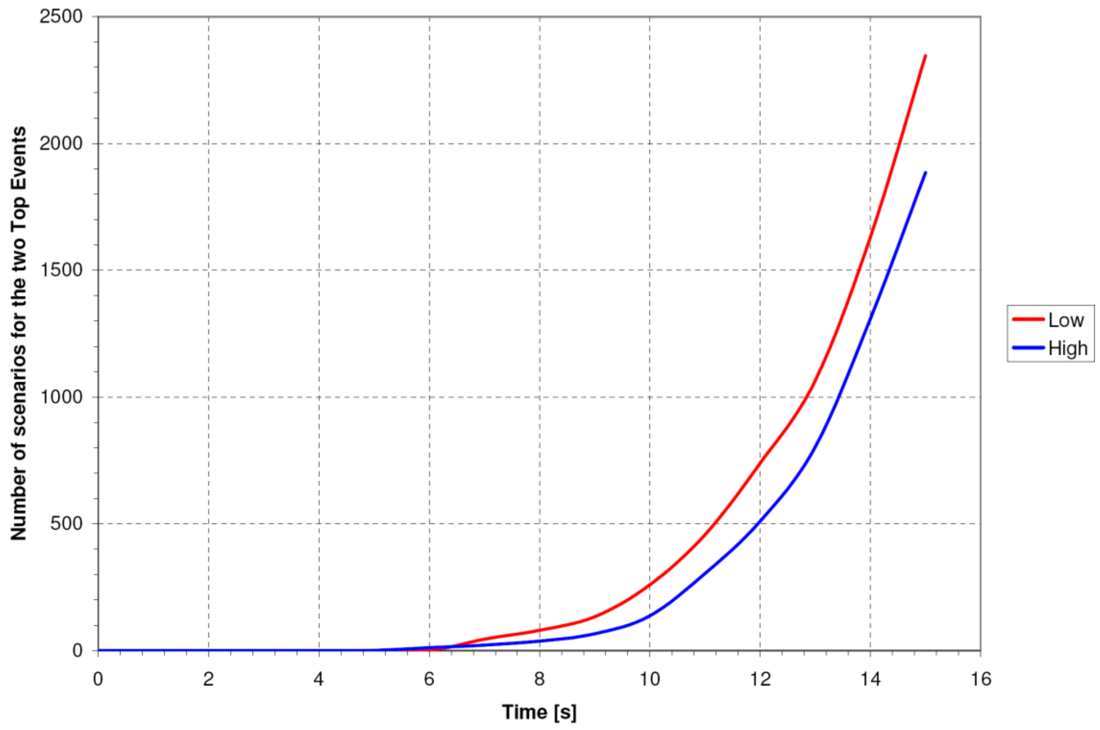


Fig.8.8 – Number of scenarios for the two Top Events generated for the EIE

Figure 8.9 shows that the exact timing of the failure of a system component can have an impact on the resulting system failure. In particular, Fig.8.9 depicts the evolution of the level variable under two distinct scenarios starting both from the same initial conditions as presented in Section 8.1 for the EIE.

In one case, the BFV fails stuck at the current position at time t = 5 sec. In the other case, the BFV fails stuck at time t = 4 sec. The first scenario results in the level failing low (x_n < - 2.0 feet), while the second scenario results in the level failing high (x_n > 2.5 feet).

This example is important because, for a system similar to the digital feedwater control system in an operating PWR, it illustrates:

- what has been reported in the literature on the possible sensitivity of the system failure mode to the exact timing of component failures [2] , and,
- that an analysis that considers only the order of events and ignores their exact timing may result in the failure to identify possible failure modes.

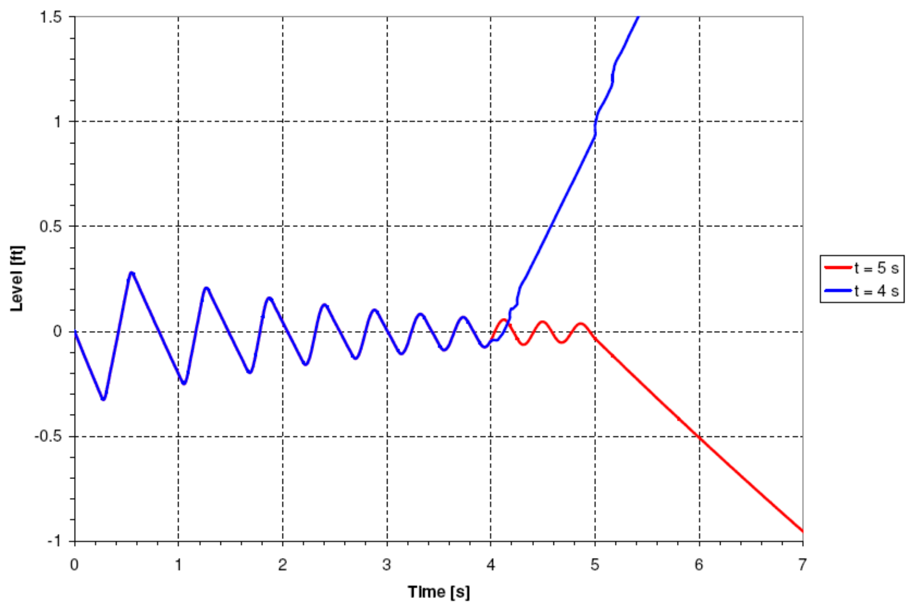


Fig.8.9 – Different Failure Modes as Result of Timing of BFV failure

Since $h(n\,|\,n', j' \rightarrow j)$ data were not available, no attempt was made to determine P_L and P_H.

CHAPTER 9

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

The methodology described in this thesis shows how it is possible to model digital control systems for PRA purposes using Markov/CCMT. In particular, it shows how it is possible to perform the analysis starting from a detailed description and understanding of the system under consideration. These inputs are the control laws, system topology, control logic and the analysis of the failures and effects performed for all the components of the system.

The analysis of the system is performed merging two separate models, one for each of the two types of interactions. Type I interactions take into account the dynamic of the system and how the process and the controller affect each other. These are modeled using the CCMT which describes the dynamic of the process through transitions between cells that compose the CVSS. Type II interactions, on the other hand, take into account the interactions among the components of the controller. These are modeled thorough Markov transitions diagrams, one for each component. Discrete hardware/software/ firmware states are defined and transitions between these states are deduced from the control logic of the system, as well as from the failure modes and effects analysis performed on each component.

Starting form a finite set of initial point distributed in the CVSS, the Markov/CCMT follows the trajectories originating from these points. At each time step and for each point, new point, having the same location in the CVSS but different system configuration based on the Markov models previously built, are generated.

The set of the overall trajectories are in principle event sequences which can be simply converted into dynamic event trees. These event sequences can be incorporated into an existing ET/FT based PRA of a PWR using the SAPHIRE code. Markov/CCMT also yields DETs that represent the ensemble average of the event sequences for the partitioning defined for the CVSS.

Thus, the objectives of this thesis presented in Section 1 are met and fulfilled. These objectives include:

1. show how it is possible to model digital control systems for PRA using the Markov/CCMT methodology, and
2. produce dynamic event trees which can be incorporated into an existing ET/FT based PRA of a PWR using the SAPHIRE code.

The methodology can capture:

1. dependence of the control action on system history,
2. dependence of system failure modes on exact timing of failures,
3. functional as well as intermittent failures,
4. error detection capability,

5. possible system recovery from failure modes

The first three requirements are satisfied by using auxiliary process variables as appropriate. For example, in the EIE, it is shown how the need to keep track of the BFV position (due to the freeze state of the controllers) is satisfied by adding the variable “BFV position”. In general, every time that the system under consideration shows history dependence properties, a new variable or a new state would need to be added. The drawback is a greater computational effort. I

This thesis also shows how the interplay of computers and controllers is modeled through a 3-layer finite state machine model. A layer shows the interactions among one or more components. The definition and the consequently construction of each layer would strongly depend on the connection network of the components themselves.

The error detection capabilities and the system recovery from failure modes of the control system are specified in the FMEA. The recovery form failure can be implemented simply adding a transition to the finite state machine model from the same state failure state to the operating state. For example, in the DFWCS, recovery from a failure in the communication (between sensors and computers or between computer and controllers) are implemented in this way.

Some areas require additional research and study. The exponential growth of the event sequences for the EIE indicates a computational challenge even for simple systems for a large number of time steps. This growth can be by either reducing the number of states of the Markov transitions diagrams or increasing the Markov time step Δt. This thesis illustrates how it is possible to reduce drastically the number of states by merging components together or states of the same Markov transition diagram. An investigation regarding the effect of the length Markov time on the overall analysis should be performed once failure rates for the failures presented in the FMEA will be available. Thus, it could be possible to observe how the Cdf and pdf of the Top Events are affected by the choice of the length of the Markov time step.

According to Perrow in [19], the DFWCS is a loosely coupled system. It is suggested to perform a Markov/CCMT analysis on tightly coupled systems [19] in order to demonstrate the feasibility of the methodology for systems with more complicated topologies.

REFERENCES

[1] “Digital Instrumentation and Control Systems in Nuclear Power Plants”, National Research Council, Nation Academy Press, Washington D.C.

[2] T. Aldemir, D. W. Miller, M. Stovsky, J. Kirschenbaum, P. Bucci, A. W. Fentiman, and L. M. Mangan, “Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments”, NUREG/CR-6901, U. S. Nuclear Regulatory Commission, Washington, D.C. (2006)

[3] S. Guarro, M. Yau, and M. Motamed, “Development of Tools for Safety Analysis of Control Software in Advanced Reactors”, NUREG/CR-6465, U.S. Nuclear Regulatory Commission, Washington, D.C. (1996)

[4] C. L. Smith, J. Knudsen, M. Calley, S. BeckK, K. Kvarfordt and S. T. Wood, “SAPHIRE Basics: An Introduction to Probability Risk Assessment Via the Systems Analysis Program for Hands-on Integrated Reliability Evaluations” SAPHIRE Software, Idaho National Laboratory, Idaho Falls, ID (2005).

[5] J. Kirscehbaum, M. Stovsky, P. Bucci, T. Aldemir and S. A. Arndt, ”Benchmark Development for Comparing Digital Instrumentation and Control System Reliability Modeling Approaches“, American Nuclear Society, LaGrange Park, IL (2005).

[6] T. Aldemir, D. Mandelli et al.,”A Benchmark System for the Assessment of Reliability Modeling Methodologies for Digital Instrumentation and Control Systems in Nuclear Plants“, Proceedings NPIC&HMIT 2006, Albuquerque, New Mexico (2006).

[7] “Reliability Modeling of Digital Instrumentation and Control Systems for nuclear Reactor Probabilistic Risk Assessments”, NUREG/CR-6942, U. S. Nuclear Regulatory Commission, Washington, D.C. (2006).

[8] J. Kirschenbaum, P. Bucci, M. Stovsky, D. Mandelli, T. Aldemir, M. Yau, S. Guarro, E. Ekici, S.A. Arndt, “A Benchmark System for Comparing Reliability Modeling Approaches for Digital Instrumentation and Control Systems”, submitted to Nuclear Technology, 2007.

[9] J. Kirschenbaum, M. Stovsky, P. Bucci, T. Aldemir, S. A. Arndt, , ”Benchmark Development for Comparing Digital Instrumentation and Control System Reliability Modeling Approaches“, American Nuclear Society, LaGrange Park, IL, 2005.

[10] B. Forouzan, “Data Communication and Networking”, 2nd Edition, Mc Graw Hill, 1991.

[11] Technical Specification Non-Compliance Due to Loose Wire on a HPCI System Valve Logic Relay, 4-03-0, (2006)

[12] J. K. Kirkwood, Reactor Trip Due to Low Steam Generator Water Level After Feed Pump Trip, 2004-01-00, (3-23-2004)

[13] N. E. Todreas and M. S. Kazimi, “Nuclear Systems: Thermal Hydraulic Fundamentals”, Hemisphere Publishing. Corp. (1990).

[14] T. Aldemir, “Computer-Assisted Markov Modeling of Process Control Systems,” IEEE Trans. Reliability, R-36, 133-144, 1987.

[15] N. J. McCormick, “Reliability and Risk Analysis: Methods and Nuclear Power Plant Applications”, Academic Press, Inc, Boston, 1981.

[16] C.S. Hsu, “Cell-to-Cell Mapping: a Method of Global Analysis for Nonlinear Systems”, Springer-Verlag, New York, 1987.

[17] T. Aldemir, “Utilization of the Cell-To-Cell Mapping Technique to Construct Markov Failure Models for Process Control Systems“, Probabilistic Safety Assessment and Management: PSAM1, 1431-1436, Elsevier, New York, 1991.

[18] T. Aldemir, D. Mandelli et al., “Incorporation of Markov Reliability Models for Digital Instrumentation and Control Systems into Existing PRAs”, Proceedings NPIC&HMIT 2006, Albuquerque, New Mexico (2006).

[19] C. Perrow, “Normal Accident, Living with High Risk Technologies”, Princeton University Press.

[20] S. Guarro, M. Yau and M. Motamed, ”Development of Tools for Safety Analysis of Control Software in Advanced Reactors”, NUREG/CR – 6465, U.S. Nuclear Regulatory Commission, Washington, D.C., 1996.

[21] P.E. Labeau, C. Smidts, S. Swaminathan, “Dynamic Reliability: Towards an Integrated Platform for Probability Risk Assessment, Reliability and System Safety”, Vol. 68, 2000.

[22] M. Hassan, T. Aldemir, ”A Data Base Oriented Dynamic Methodology for the Failure Analysis of Closed Loop Control Systems in Process Plants“, Reliability Engineering & System Safety, Vol. 27, 275-322, February 1990.

[23] B. Tombuyses, T. Aldemir, ”Continuous Cell-to-Cell Mapping“, J. Sound and Vibration, Vol. 202(3), 395-415, May 1997.

[24] M. Belhadj, T. Aldemir, “Some Computational Improvements in Process System Reliability and Safety Analysis Using Dynamic Methodologies”, Reliability and System Safety, Vol. 52, 1996.

APPENDIX A

PROGRAM

The program which implements the Markov/CCMT methodology for the EIE is presented in the following pages.

```
import java.util.Scanner;
import java.io.*;
import java.util.Vector;
import java.lang.Math;

import flanagan.integration.RungeKutta;

public class EIE
{
      public static void main(String[] args) throws IOException
     {
            double [][] matrix_n =
            {
                        {1,1,1,1,1,1,1},
                        {0,0,1,1,1,1,1},
                        {0,0,0,1,1,1,1},
                        {0,0,0,1,1,1,1},
                        {0,0,0,0,1,1,1},
                        {0,0,0,0,0,1,1},
                        {0,0,0,0,0,0,1},
            };


            int simulationtime = 2;  // in seconds
            int dt = 1;
            int timesteps = simulationtime/dt;

            Vector<branch> tree = new Vector<branch>(0,1);

            branch zero = new branch (timesteps);

            tree.add(0, zero);

            branch temp;      // temp e' il branch sotto considerazione

            for (int i=0 ; i<tree.capacity() ; i++)               {
                  temp = tree.elementAt(i);

                  for (int j=temp.completedSteps-1 ; j<timesteps-1 ;
j++)                    {
                        tree = nodesAnalysis (tree, i, j, matrix_n,
dt);
                  }
            }

            PrintWriter outputFile = new PrintWriter (new
FileWriter("c:\\eie.csv"));

            branch temp2;

            for (int y = 0 ; y < tree.capacity() ; y++)
      {
```

```
                    temp2 = tree.elementAt(y);

                    for (int x = 0 ; x < timesteps ; x++)
                    {
                          outputFile.print(temp2.path[x].n + " , " );
                    }

                    outputFile.println();
        }

              outputFile.close();

              System.out.println(tree.capacity());

              branch temp3;

              int failH = 0;
              int failL = 0;
              int OK = 0;

              for (int y = 0 ; y < tree.capacity() ; y++)
              {
                    boolean H = false;
                    boolean L = false;
                    boolean sistemOK = true;

                    temp3 = tree.elementAt(y);

                    for (int x = 0 ; ((x<timesteps) && sistemOK) ; x++)
                    {
                          if (temp3.path[x].l<(-2))
                          {
                                L = true;
                                sistemOK = false;
                                failL++;
                          }
                          if (temp3.path[x].l>2.5)
                          {
                                H = true;
                                sistemOK = false;
                                failH++;
                          }
                    }
              }

              System.out.println(tree.capacity() + " , " + failL + " , "
+ failH);

      }


        public static step nextStep (step initial, int t, int deltatau,
int n)
        {
              step fin = new step(n, initial.l, initial.cl, initial.le,
  initial.bfvpos);

              DifferentialEquations des =
  DifferentialEquations.getInstance();

          if (initial.n == 0) // CV
          {
              des.setUseOldBfvPosition(false);
          }
          if (initial.n == 1) // OV
          {
              des.setUseOldBfvPosition(true);
              des.setBfvPosition(initial.bfvpos);
          }
          if (initial.n == 2) // OV
          {
              des.setUseOldBfvPosition(true);
              des.setBfvPosition(initial.bfvpos);
          }
          if (initial.n == 3) // OV
          {
              des.setUseOldBfvPosition(true);
              des.setBfvPosition(initial.bfvpos);
          }
          if (initial.n == 4) // AO
          {
              des.setUseOldBfvPosition(false);
              des.setBfvPosition(Math.random()*100);
          }
          if (initial.n == 5) // 0 vdc
          {
              des.setUseOldBfvPosition(false);
              des.setBfvPosition(0);
          }
          if (initial.n == 6) // OV
          {
              des.setUseOldBfvPosition(true);
              des.setBfvPosition(initial.bfvpos);
          }

              double[] X = RungeKutta.fourthOrder(des, t , new double[] {
  initial.l , initial.le , initial.cl }, t + deltatau, 0.01 * deltatau);

              fin.n = n;
          fin.l  = X[0];
          fin.le = X[1];
          fin.cl = X[2];
          fin.bfvpos = clamp(des.getBfvPosition(), 0.0, 100.0);
          fin.bfvpos = des.getBfvPosition();

              return fin;
        }
```

```
	public static Vector<branch> nodesAnalysis (Vector<branch> 
oldtree, int row, int column, double [][] g, int dt)
	{
		branch origin = oldtree.elementAt(row);

		int combination = origin.path[column].n;

		int count = 0;

		for (int l=0 ; l<7 ; l++)
		{
			if (g [combination][l] == 1)
			{
				count++;

				if (count == 1)
				{
					origin.path[column+1] = 
nextStep(origin.path[column], column*dt, dt, l);

					origin.completedSteps++;

					oldtree.removeElementAt(row);

					oldtree.add(row,origin);
				}
				else // crea una nuova branch
				{
					oldtree = branchGeneration(oldtree, row, 
column, l, dt);
				}
			}
		}
		return oldtree;
	}


	public static Vector<branch> branchGeneration (Vector<branch> ot, 
int r, int c, int stateComb, int t)
	{
		branch or = new branch (ot.elementAt(r).path.length);

		or.completedSteps = ot.elementAt(r).completedSteps;

		for (int i=0; i<ot.elementAt(r).completedSteps ; i++)
		{
			or.path[i] = new step (ot.elementAt(r).path[i].n , 
ot.elementAt(r).path[i].l , ot.elementAt(r).path[i].cl, 
ot.elementAt(r).path[i].le, ot.elementAt(r).path[i].bfvpos);
		}

		int d = or.completedSteps-1;

		or.path[d].n = stateComb;
```

```
            or.path[d] = nextStep(or.path[d] , c*t , t , stateComb);

            ot.add(or);

            return ot;
      }

      public static double clamp(double currBfvPos, double min, double
max)
    {
          if (currBfvPos < min)
          {
            return min;
          }
          else if (currBfvPos > max)
          {
            return max;
          }
          else
          {
            return currBfvPos;
          }
    }

}
```